Integrator™/CM920T-ETM

User Guide

ARM

Copyright © ARM Limited 2001. All rights reserved.
ARM DUI 0149A

Integrator/CM920T-ETM
User Guide

Copyright © ARM Limited 2001. All rights reserved.

Release Information

Description	Issue	Change
21 February 2001	A	New document.

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks owned by ARM Limited, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Conformance Notices

This section contains conformance notices.

Federal Communications Commission Notice

This device is test equipment and consequently is exempt from part 15 of the FCC Rules under section 15.103 (c).

CE Declaration of Conformity

This equipment has been tested according to ISE/IEC Guide 22 and EN 45014. It conforms to the following product EMC specifications:

The product herewith complies with the requirements of EMC Directive 89/336/EEC as amended.

Confidentiality Status

This document is Open Access. This document has no restriction on distribution.

Product Status

The information in this document is final (information on a developed product).

Web Address

http://www.arm.com

Copyright © ARM Limited 2001. All rights reserved.

Copyright © ARM Limited 2001. All rights reserved.

Contents
Integrator/CM920T-ETM User Guide

Preface
About this document viii
Further reading x
Feedback xi

Chapter 1 Introduction
1.1 About the core module 1-2
1.2 Core module architecture 1-4
1.3 Links and indicators 1-8
1.4 Test points 1-10
1.5 Precautions 1-12

Chapter 2 Getting Started
2.1 Setting up a standalone core module 2-2
2.2 Attaching the core module to a motherboard 2-5

Chapter 3 Hardware Description
3.1 ARM microprocessor test chip 3-2
3.2 SSRAM controller 3-3
3.3 Core module FPGA 3-4
3.4 SDRAM controller 3-6
3.5 Reset controller 3-8

Copyright © ARM Limited 2001. All rights reserved.

3.6 System bus bridge 3-11
3.7 Module ID selection 3-19
3.8 Clock generators 3-21
3.9 Multi-ICE support 3-26
3.10 Embedded Trace support 3-35

Chapter 4 Programmer's Reference
4.1 Memory organization 4-2
4.2 Exception vector mapping 4-8
4.3 Core module registers 4-9
4.4 Core module flag registers 4-22
4.5 Core module interrupt registers 4-23
4.6 SDRAM SPD memory 4-27

Appendix A Signal Descriptions
A.1 HDRA A-2
A.2 HDRB A-4
A.3 Trace connector pinout A-10
A.4 Logic analyzer connectors A-11

Appendix B Specifications
B.1 Electrical specification B-2
B.2 Timing specification B-3
B.3 Mechanical details B-11

Index

 Copyright © ARM Limited 2001. All rights reserved.

Preface

This preface introduces the ARM Integrator/CM920T-ETM core module and associated reference documentation. It contains the following sections:

- *About this document* on page viii
- *Further reading* on page x
- *Feedback* on page xi.

Copyright © ARM Limited 2001. All rights reserved.

About this document

This document describes how to set up and use the ARM Integrator/CM920T-ETM core module.

Intended audience

This document has been written for experienced hardware and software developers to aid the development of ARM-based products using the core module as part of a development system.

Organization

This document is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an introduction to the core module. This chapter shows the physical layout of the core module and identifies the main components.

Chapter 2 ***Getting Started***

Read this chapter for a description of how to set up and start using the core module. This chapter describes how to connect the core module and how to apply power.

Chapter 3 ***Hardware Description***

Read this chapter for a description of the hardware architecture of the core module. This chapter describes the clocks, resets, and debug hardware provided by the core module.

Chapter 4 ***Programmer's Reference***

Read this chapter for a description of the core module memory map and registers.

Appendix A ***Signal Descriptions***

Refer to this appendix for a description of the signals on the HDRA and HDRB connectors.

Appendix B ***Specifications***

Refer to this appendix for electrical, timing, and mechanical specifications.

Copyright © ARM Limited 2001. All rights reserved.

Typographical conventions

The following typographical conventions are used in this book:

`typewriter` Denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.

`type`writer Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.

`typewriter italic`

Denotes arguments to commands and functions where the argument is to be replaced by a specific value.

italic Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

bold Highlights interface elements, such as menu names and buttons. Also used for terms in descriptive lists, where appropriate.

`typewriter bold`

Denotes language keywords when used outside example code and ARM processor signal names.

Copyright © ARM Limited 2001. All rights reserved.

Further reading

This section lists related publications by ARM Limited and other companies provide additional information.

ARM publications

The following publications provide information about related ARM products and toolkits:

- *ARM920T Technical Reference Manual* (ARM DDI 0151)
- *ARM Integrator/AP User Guide* (ARM DUI 0098)
- *ARM Integrator/SP User Guide* (ARM DUI 0099)
- *ARM Multi-ICE User Guide* (ARM DUI 0048)
- *AMBA Specification* (ARM IHI 0011)
- *ARM Architectural Reference Manual* (ARM DDI 0100)
- *ARM Firmware Suite Reference Guide* (ARM DUI 0102)
- *ARM Software Development Toolkit User Guide* (ARM DUI 0040)
- *ARM Software Development Toolkit Reference Guide* (ARM DUI 0041)
- *ADS Tools Guide* (ARM DUI 0067)
- *ADS Debuggers Guide* (ARM DUI 0066)
- *ADS Debug Target Guide* (ARM DUI 0058)
- *ADS Developer Guide* (ARM DUI 0056)
- *ADS CodeWarrior IDE Guid*e (ARM DUI 0065).

Other publications

The following publication provides information about the clock controller chip used on the Integrator modules:

- *MicroClock OSCaR User Configurable Clock Data Sheet* (MDS525), MicroClock Division of ICS, San Jose, CA.

The following publications provide information and guidelines for developing products for Microsoft Windows CE:

- *Standard Development Board for Microsoft® Windows® CE*, 1998, Microsoft Corporation
- *HARP Enclosure Requirements for Microsoft® Windows® CE*, 1998, Microsoft Corporation.

Further information on these topics is available from the Microsoft web site.

Copyright © ARM Limited 2001. All rights reserved.

Feedback

ARM Limited welcomes feedback both on the ARM Integrator/CM920T-ETM core module and on the documentation.

Feedback on this document

If you have any comments about this document, please send email to `errata@arm.com` giving:

- the document title
- the document number
- the page number(s) to which your comments refer
- an explanation of your comments.

General suggestions for additions and improvements are also welcome.

Feedback on the ARM Integrator/CM920T-ETM

If you have any comments or suggestions about this product, please contact your supplier giving:

- the product name
- an explanation of your comments.

Copyright © ARM Limited 2001. All rights reserved.

 Copyright © ARM Limited 2001. All rights reserved.

Chapter 1
Introduction

This chapter introduces the ARM Integrator/CM920T-ETM and core module. It contains the following sections:

- *About the core module* on page 1-2
- *Core module architecture* on page 1-4
- *Links and indicators* on page 1-8
- *Test points* on page 1-10
- *Precautions* on page 1-12.

Copyright © ARM Limited 2001. All rights reserved.

1.1 About the core module

The Integrator/CM920T-ETM core module provides you with a development system which can be used to develop products around the ARM920T-ETM core.

The core module can be used in a number of different ways. With power and a connection to a Multi-ICE® unit, the core module provides a basic development system. By mounting the core module onto an Integrator motherboard or other Integrator modules, you can build a realistic emulation of the system being developed.

The core module can be used in the following ways:

- as a standalone development system
- mounted onto an ARM Integrator motherboard
- mounted onto an ARM logic module without a motherboard, with the logic module providing the system controller functions of a motherboard.
- integrated into a third-party development or ASIC prototyping system.

Figure 1-1 on page 1-3 shows the layout of the ARM Integrator/CM920T-ETM.

Copyright © ARM Limited 2001. All rights reserved.

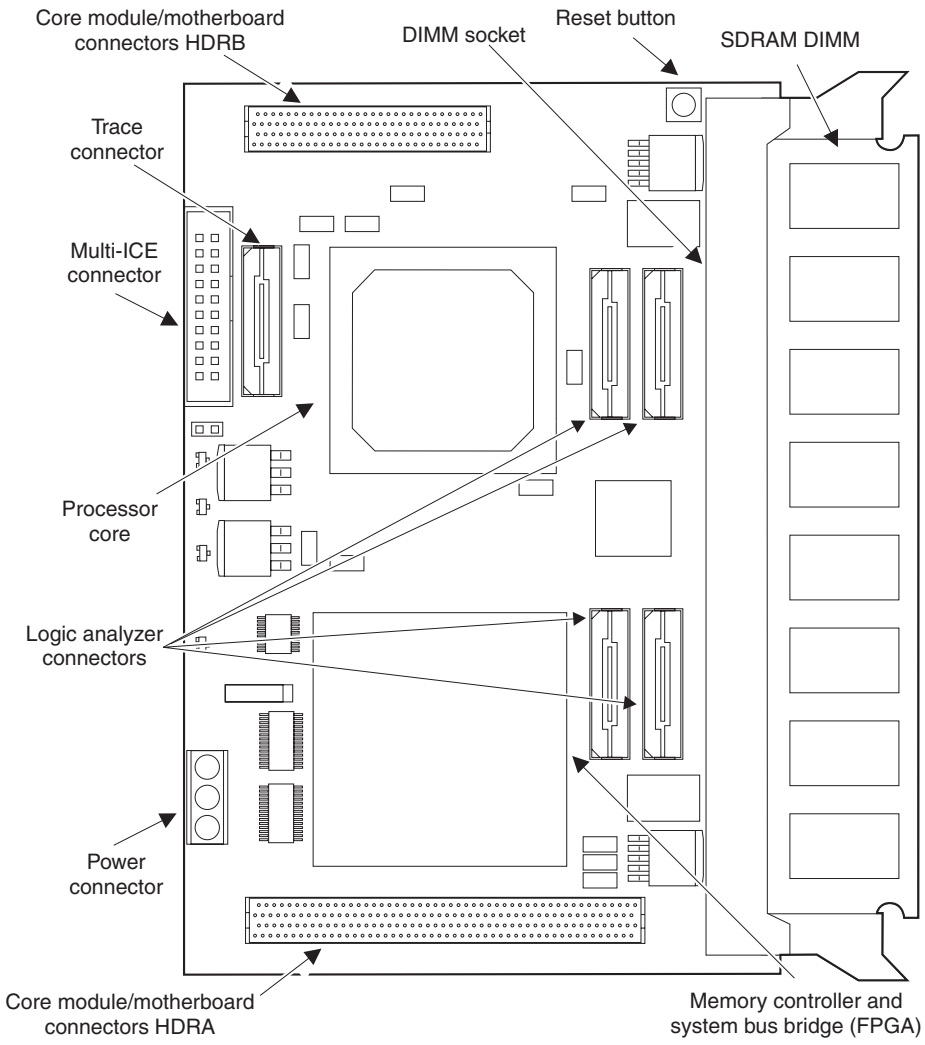


Figure 1-1 Integrator/CM920T-ETM layout

Copyright © ARM Limited 2001. All rights reserved.

1.2 Core module architecture

The major components on the core module are as follows:

- ARM920T microprocessor core with *Embedded Trace Module* (ETM)
- core module FPGA that implements:
  - SDRAM controller
  - system bus bridge
  - reset controller
  - interrupt controller
  - status, configuration, and interrupt registers.
- 1MB SSRAM.
- up to 256MB of SDRAM (optional) plugged into the DIMM socket
- SSRAM controller
- clock generator
- system bus connectors
- logic analyzer connectors for AHB and Trace port.

1.2.1 System architecture

Figure 1-2 illustrates the architecture of the core module.

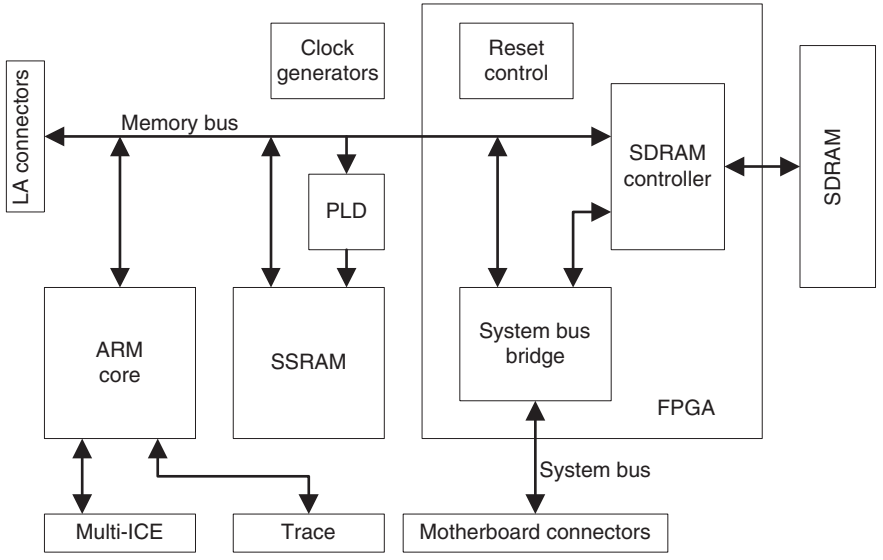


Figure 1-2 ARM Integrator/CM920T-ETM block diagram

Copyright © ARM Limited 2001. All rights reserved.

1.2.2 ARM processor test chip

The Integrator/CM920T-ETM is be fitted with an ARM920T core. For a brief description of this core, see *ARM microprocessor test chip* on page 3-2.

1.2.3 Core module FPGA

The FPGA provides system control functions for the core module, enabling it to operate as a standalone development system or attached to a motherboard. These functions are outlined in this section and described in detail in Chapter 3 *Hardware Description*.

SDRAM controller

The SDRAM controller is implemented within the FPGA. This provides support for *Dual In-line Memory Modules* (DIMMs) with a capacity of between 16 and 256MB. See *SDRAM controller* on page 3-6.

Reset controller

The reset controller initializes the core. The core module can be reset from five sources:

- reset button
- motherboard
- other core modules
- Multi-ICE
- software.

For information about the reset controller, see *Reset controller* on page 3-8.

System bus bridge

The system bus bridge provides an AMBA interface between the memory bus on the core module and the system bus on a motherboard. It allows the processor to access resources on the motherboard and on other modules. It also allows other masters to access the core module SDRAM (see *System bus bridge* on page 3-11).

Status and configuration space

The status and configuration space contains status and configuration registers for the core module. These provide the following information and control:

- type of processor and whether it has a cache, MMU, or protection unit
- the position of the core module in a multi-module stack
- SDRAM size, address configuration, and CAS latency setup

Copyright © ARM Limited 2001. All rights reserved.

- core module oscillator setup
- interrupt control for the processor debug communications channel.

The status and control registers can only be accessed by the local processor. For more information about the status and control registers see Chapter 4 *Programmer's Reference*.

1.2.4 Volatile memory

The volatile memory system includes an SSRAM device, and a plug-in SDRAM memory module (referred to as *local* SDRAM when it is on the same core module as the processor). These areas of memory are closely coupled to the processor core to ensure high performance. The core module uses separate memory and system buses to avoid memory access performance being degraded by bus loading.

The SDRAM controller is implemented within the core module controller FPGA and a separate SSRAM controller is implemented with a *Programmable Logic Device* (PLD).

The SDRAM can be accessed by the local processor, by processors on other core modules, and by other system bus masters.

The SSRAM can only be accessed by the local processor.

1.2.5 Clock generator

The core module uses four clock signals:

REFCLK A fixed frequency 24MHz signal that can be used by the FPGA to generate real-time delays.

CORECLK A programmable frequency clock input to the ARM test chip.

LCLK A programmable frequency clock for the local memory bus.

AUXCLK A programmable frequency clock reserved for future use.

The programmable clocks are supplied by three clock generator chips. Their frequencies are selected in oscillator control registers within the FPGA. A reference clock is supplied to the clock generators and to the FPGA (see *Clock generators* on page 3-21). The memory bus and system bus are asynchronous, allowing each to be run at the speed of its slowest device without compromising the performance of other buses in the system.

 Copyright © ARM Limited 2001. All rights reserved.

1.2.6 Multi-ICE connector

The Multi-ICE connector enables JTAG hardware debugging equipment, such as Multi-ICE, to be connected to the core module. It is possible to both drive and sense the system-reset line (**nSRST**), and to drive JTAG reset (**nTRST**) to the core from the Multi-ICE connector. See *Multi-ICE support* on page 3-26.

Copyright © ARM Limited 2001. All rights reserved.

1.3 Links and indicators

The core module provides one link and four surface-mounted LEDs. These are illustrated in Figure 1-3.

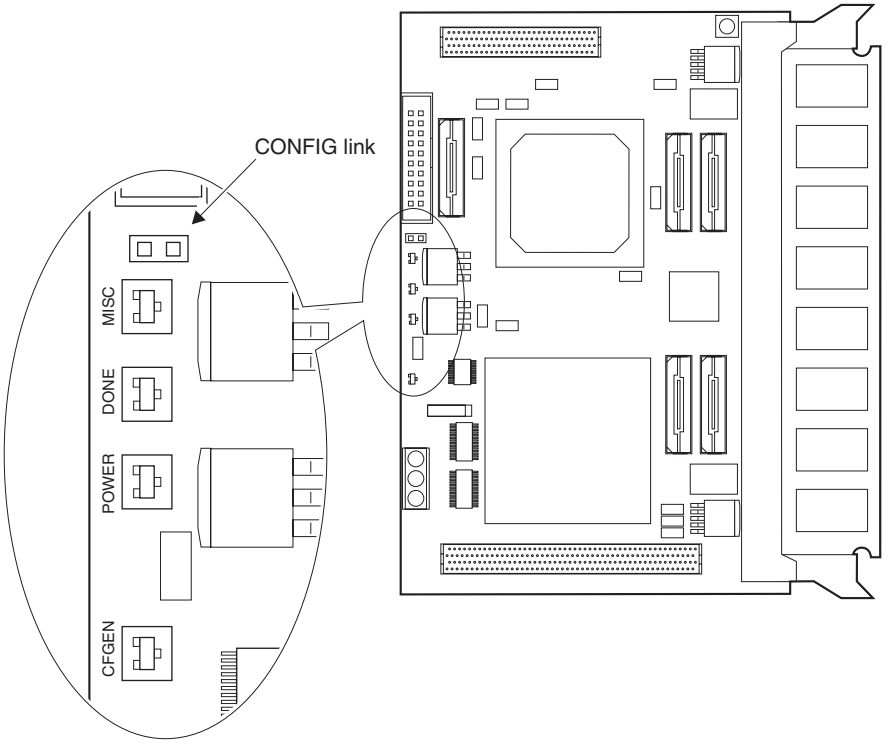


Figure 1-3 Links and indicators

1.3.1 CONFIG link

The core module has only one link, marked CONFIG. This is left open during normal operation. It is only fitted when downloading new FPGA and PLD configuration information.

Copyright © ARM Limited 2001. All rights reserved.

1.3.2 LED indicators

The functions of the four surface-mounted LEDs are summarized in Table 1-1.

Table 1-1 LED functional summary

Name	Color	Function
MISC	Green	This LED is controlled using the control register (see *Core module control register* on page 4-13).
DONE	Green	This LED illuminates when the FPGA has successfully loaded its configuration information following power-on.
POWER	Green	This LED illuminates to indicate that a 3.3V supply is present.
CFGLED	Orange	This LED illuminates to indicate that the CONFIG link is fitted.

Copyright © ARM Limited 2001. All rights reserved.

1.4 Test points

The core module provides test points and ground points to aid diagnostics. The most useful of these are illustrated in Figure 1-4.

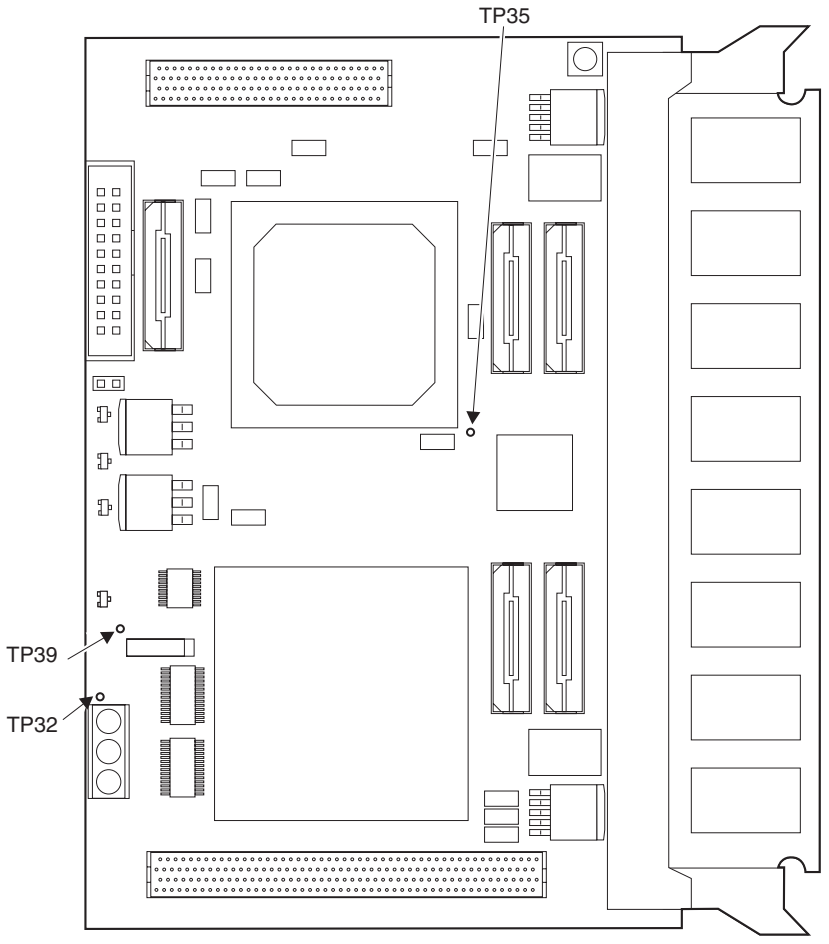


Figure 1-4 Test points

The functions of these test points are summarized in Table 1-2 on page 1-11. For information about setting the frequency of the core clock and auxiliary clock, see *Clock generators* on page 3-21.

Copyright © ARM Limited 2001. All rights reserved.

Table 1-2 Test point functions

Test point	Name	Function
TP35	FCLKOUT	Clock output from the microprocessor core
TP32	AUXCLK	Auxiliary clock
TP39	REFCLK	Reference clock (24MHz)

Copyright © ARM Limited 2001. All rights reserved.

1.5 Precautions

This section contains safety information and advice on how to avoid damage to the core module.

1.5.1 Ensuring safety

The core module is powered from 3.3V and 5V DC supplies.

Warning

To avoid a safety hazard, only connect *Safety Extra Low Voltage* (SELV) equipment to the JTAG interface.

1.5.2 Preventing damage

The core module is intended for use within a laboratory or engineering development environment. It is supplied without an enclosure which leaves the board sensitive to electrostatic discharges and allows electromagnetic emissions.

Caution

To avoid damage to the board, observe the following precautions.

- never subject the board to high electrostatic potentials.
- always wear a grounding strap when handling the board.
- only hold the board by the edges.
- avoid touching the component pins or any other metallic element.

Caution

Do not use the board near equipment which is:

- sensitive to electromagnetic emissions (such as medical equipment)
- a transmitter of electromagnetic emissions.

Copyright © ARM Limited 2001. All rights reserved.

Chapter 2
Getting Started

This chapter describes how to set up and prepare the core module for use. It contains the following sections:

- *Setting up a standalone core module* on page 2-2
- *Attaching the core module to a motherboard* on page 2-5.

Copyright © ARM Limited 2001. All rights reserved.

2.1 Setting up a standalone core module

To set up the core module as a standalone development system:

1. Optionally, fit an SDRAM DIMM.
2. Supply power.
3. Connect Multi-ICE.

2.1.1 Fitting an SDRAM DIMM

You should fit the following type of SDRAM module:

- PC66, PC100, or PC133- compliant 168-pin DIMM
- unbuffered
- 3.3V
- 16MB, 32MB, 64MB, 128MB or 256MB.

To install an SDRAM DIMM:

1. Ensure that the core module is powered down.
2. Open the SDRAM retaining latches outwards.
3. Press the SDRAM module into the edge connector until the retaining latches click into place.

Note

The DIMM edge connector has polarizing notches to ensure that it is correctly oriented in the socket.

2.1.2 Using the core module without SDRAM

You can operate the core module without SDRAM because it has 1MB of SSRAM permanently fitted. When using ADW or AXD, you can adjust the `top_of_memory` internal variable from its default value to `0x100000`.

For further information about ARM debugger internal variables, refer to the *Software Development Toolkit Reference Guide* or *ADS Debuggers Guide*.

 Copyright © ARM Limited 2001. All rights reserved.

2.1.3 Supplying power

When using the core module as a standalone development system, you should connect a bench power supply with 3.3V and 5V outputs to the power connector, as illustrated in Figure 2-1.

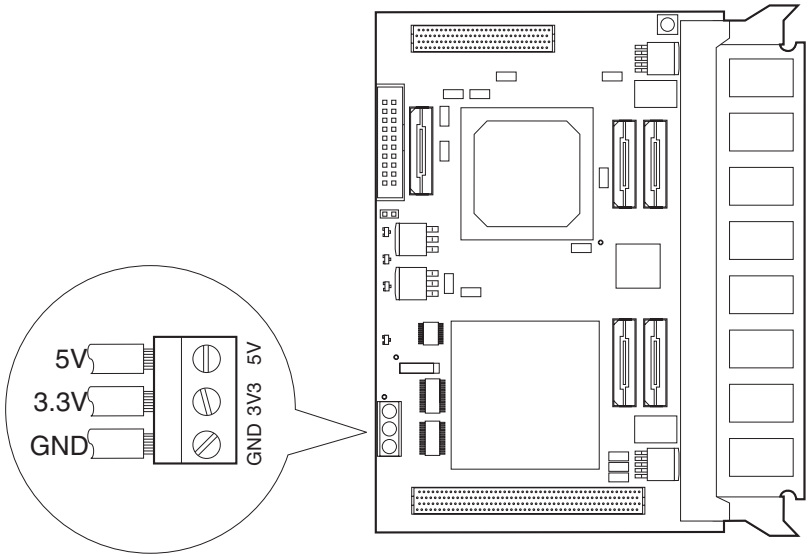


Figure 2-1 Power connector

Note

This power connection is not required when the core module is fitted to a motherboard.

Copyright © ARM Limited 2001. All rights reserved.

2.1.4 Connecting Multi-ICE

When you are using the core module as a standalone system, Multi-ICE debugging equipment can be used to download programs. The Multi-ICE setup for a standalone core module is shown in Figure 2-2.

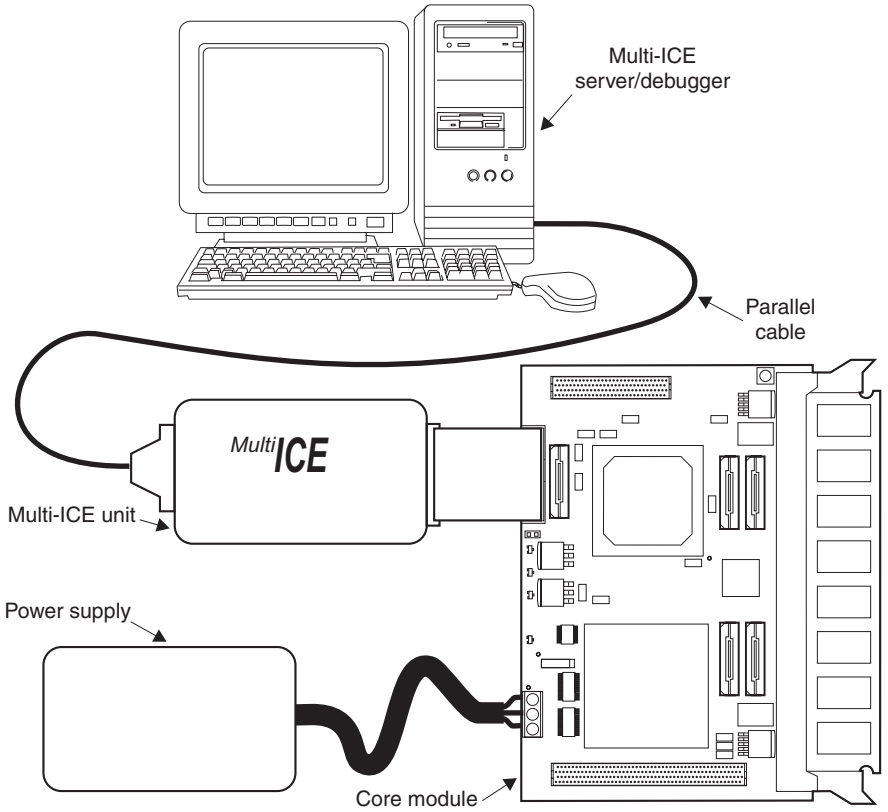


Figure 2-2 Multi-ICE connection to a core module

Caution

Because the core module does not provide nonvolatile memory, programs are lost when the power is removed.

Multi-ICE can also be used when a core module is attached to a motherboard. If more than one core module is attached, then the Multi-ICE unit must be connected to the module at the top of the stack. The Multi-ICE server and the debugger can be on one computer or on two networked computers.

Copyright © ARM Limited 2001. All rights reserved.

2.2 Attaching the core module to a motherboard

Attach the core module onto a motherboard (for example, the ARM Integrator/SP) by engaging the connectors HDRA and HDRB on the bottom of the core module with the corresponding connectors on the top of the motherboard. The lower side of the core module has sockets and the upper side of the core module has plugs to allow core modules to be mounted on top of one another. A maximum of four core modules can be stacked on a motherboard.

Figure 2-3 illustrates an example development system with four core modules attached to an ARM Integrator/SP motherboard.

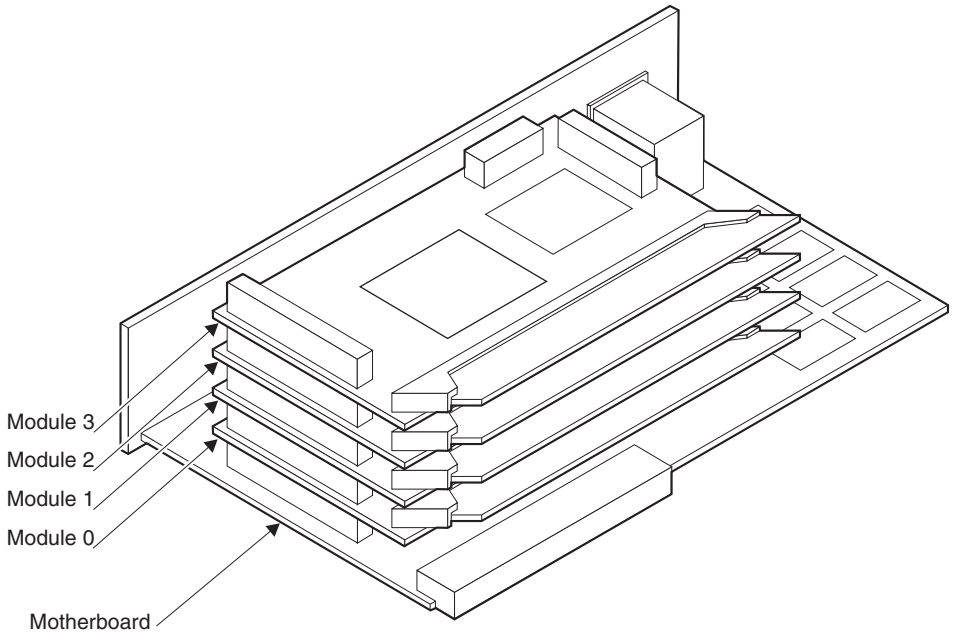


Figure 2-3 Assembled Integrator system

Note

For correct operation of the core module, do not use the core module in the EXPA/EXPB stack position on the Integrator/AP.

Copyright © ARM Limited 2001. All rights reserved.

2.2.1 Core module ID

The ID of the core module is configured automatically by the connectors (there are no links to set) and depends on its position in the stack:

- core module 0 is installed first
- core module 1 is installed next, and cannot be fitted without core module 0
- core module 2 is installed next, and cannot be fitted without core module 1
- core module 3 is installed next, and cannot be fitted without core module 2.

The ID of the core module also defines the ID of the microprocessor it carries and the system bus address of its SDRAM. The mechanism that controls the ID and mapping of the core module is described in *Module ID selection* on page 3-19. The position of a core module in the stack can be read from the CM_STAT register (see *Core module status register* on page 4-14).

2.2.2 Powering the assembled Integrator development system

Power the assembled Integrator development system by:

- connecting a bench power supply to the motherboard
- installing the motherboard in a card cage or an ATX-type PC case, depending on type.

For further information, refer to the user guide for the motherboard you are using.

Copyright © ARM Limited 2001. All rights reserved.

Chapter 3
Hardware Description

This chapter describes the on-board hardware. It contains the following sections:

- *ARM microprocessor test chip* on page 3-2
- *SSRAM controller* on page 3-3
- *Core module FPGA* on page 3-4
- *SDRAM controller* on page 3-6
- *Reset controller* on page 3-8
- *System bus bridge* on page 3-11
- *Module ID selection* on page 3-19
- *Clock generators* on page 3-21
- *Multi-ICE support* on page 3-26
- *Embedded Trace support* on page 3-35.

Copyright © ARM Limited 2001. All rights reserved.

3.1 ARM microprocessor test chip

The ARM920T core is a member of the ARM9 Thumb family of processor cores. It incorporates the following features:

- ARM9TDMI 32-bit RISC core
- Harvard architecture
- Thumb 16-bit compressed instruction set for higher code density
- JTAG interface to EmbeddedICE logic
- AMBA bus interface.
- System-on-a-chip integration features for embedded production test.

The ARM9TDMI core executes both the 32-bit ARM and 16-bit Thumb instruction sets, allowing you to switch between high performance operation and high code density. The ARM9TDMI is user code binary-compatible with ARM7TDMI, ARM10TDMI, and StrongARM processors, and is supported by a wide range of tools, operating systems, and application software.

In addition to the ARM9TDMI core, the ARM920T processor features:

- separate 16KB instruction and data caches
- write buffer
- *Memory Management Unit* (MMU)
- instruction and data *Translation Lookaside Buffers* (TLBs)
- fast context switching and high vector extensions
- 1KB tiny page mapping
- AMBA bus interface.

The instruction and data caches provide write-through or write-back operation under software control on a per-region basis.

The MMU features separate instruction and data TLBs. It supports tiny page mapping to allow large number small objects to be implemented as required by some operating systems, such as WindowsCE. WindowsCE is also supported by fast context switching and high vector extensions.

For more information about the ARM920T, refer to the *ARM920T Technical Reference Manual*.

 Copyright © ARM Limited 2001. All rights reserved.

3.2 SSRAM controller

The SSRAM controller is implemented in a Xilinx 9572XL PLD which enables the SSRAM to achieve single-cycle operation. In addition to controlling accesses to the SSRAM, the controller generates the processor response signals (**BWAIT**, **BERROR**, **BLAST**) for all accesses to:

- SSRAM
- SDRAM
- status and configuration register space
- system bus bridge.

Copyright © ARM Limited 2001. All rights reserved.

3.3 Core module FPGA

The core module FPGA contains five main functional blocks:

- *SDRAM controller* on page 3-6
- *Reset controller* on page 3-8
- *System bus bridge* on page 3-11
- *Debug communications interrupts* on page 3-34
- *Core module registers* on page 4-9.

The FPGA provides sufficient functionality for the core module to operate as a standalone development system, although with limited capabilities. System bus arbitration, system interrupt control, and input/output resources are provided by the system controller FPGA on the motherboard. See the user guide for your motherboard for further information.

Figure 3-1 illustrates the function of the core module FPGA and shows how it connects to the other devices in the system.

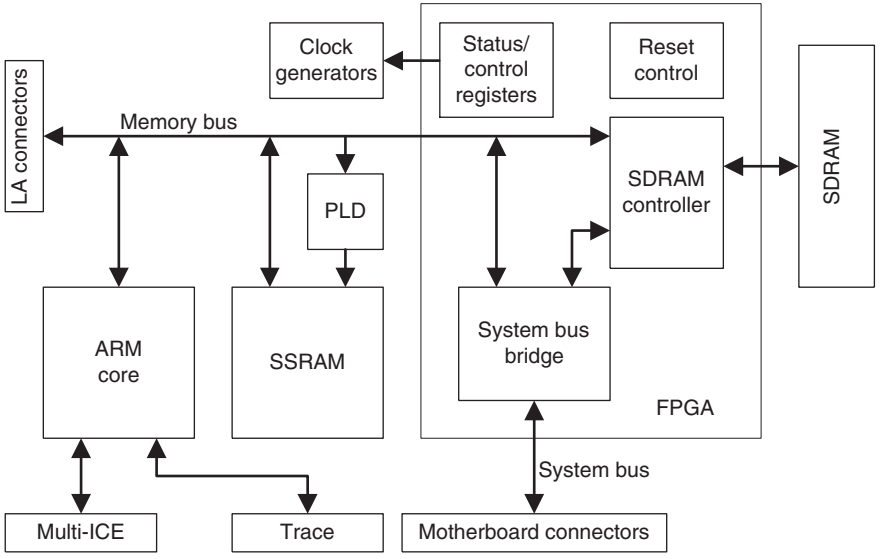


Figure 3-1 FPGA functional diagram

At power-up the FPGA loads its configuration data from a flash memory device. Parallel data from the flash is serialized by the *Programmable Logic Device* (PLD) into the configuration inputs of the FPGA. Figure 3-2 on page 3-5 shows the FPGA configuration mechanism.

Copyright © ARM Limited 2001. All rights reserved.

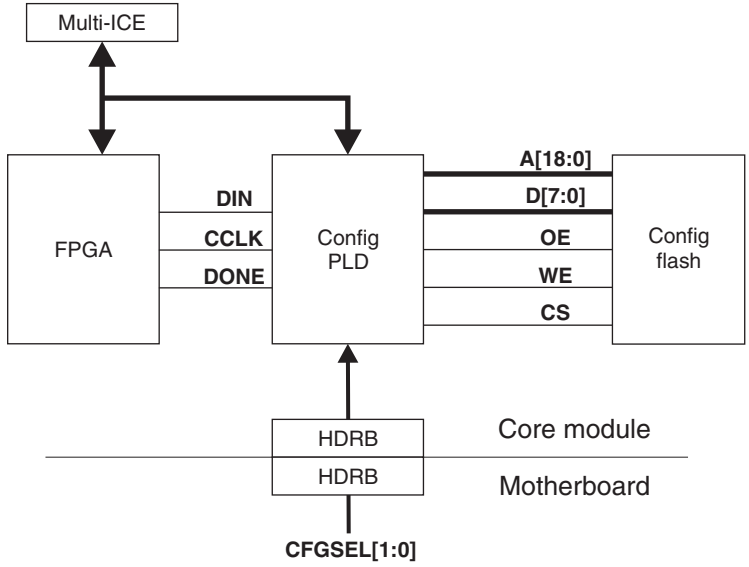


Figure 3-2 FPGA configuration

The config flash contains multiple images that enable the FPGA to be configured to support an AHB or ASB motherboard. Image selection is controlled by the static configuration select signals **CFGSEL[1:0]** from the motherboard. The encoding of these signals is shown in Table 3-1.

Table 3-1 CFGSEL[1:0] encoding

CFGSEL[1:0]	Description
00	Little endian ASB
01	Reserved
10	Little endian AHB
11	Reserved

Multi-ICE can be used to reprogram the PLD, FPGA, and flash when the core module is placed in configuration mode. See *Multi-ICE support* on page 3-26.

Copyright © ARM Limited 2001. All rights reserved.

3.4 SDRAM controller

The core module provides support for a single 16, 32, 64, 128, or 256MB SDRAM DIMM.

3.4.1 SDRAM operating mode

The operating mode of the SDRAM devices is controlled with the mode set register within each SDRAM. These registers are set immediately after power-up to specify:

- a burst size of four for both reads and writes
- *Column Address Strobe* (CAS) latency of 2 cycles.

You can program the CAS latency and memory size using the SDRAM control register (CM_SDRAM) at address `0x10000020`. See *SDRAM status and control register* on page 4-19.

Note

Before the SDRAM is used, it is necessary to read the SPD memory and program the CM_SDRAM register with the parameters indicated in Table 4-9 on page 4-19. If these values are not correctly set then SDRAM accesses might be slow or unreliable.

3.4.2 Access arbitration

The SDRAM controller provides two ports to support reads and writes by the local processor core and by masters on the system bus. The SDRAM controller uses an alternating priority scheme to ensure that the processor core and other system bus masters have equal access (see *System bus bridge* on page 3-11).

3.4.3 Serial presence detect

JEDEC-compliant SDRAM DIMMs incorporate a *Serial Presence Detect* (SPD) feature. This comprises a 2048-bit serial EEPROM located on the DIMM with the first 128 bytes programmed by the DIMM manufacturer to identify the following:

- module type
- memory organization
- timing parameters.

The EEPROM clock (SCL) operates at 93.75kHz (24MHz divided by 256). The transfer rate for read accesses to the EEPROM is 100kbit/s maximum. The data is read out serially 8 bits at a time, preceded by a start bit and followed by a stop bit. This makes reading the EEPROM a very slow process because it takes approximately 27ms to read all 256 bytes. However, during power-up the contents of the EEPROM are copied into

 Copyright © ARM Limited 2001. All rights reserved.

a 64 x 32-bit area of memory (CM_SPD) within the SDRAM controller. The SPD flag is set in the SDRAM control register (CM_SDRAM) when the SPD data is available. This copy can be randomly accessed at `0x10000100` to `0x100001FC` (see *SDRAM SPD memory* on page 4-27).

Write accesses to the SPD EEPROM are not supported.

Copyright © ARM Limited 2001. All rights reserved.

3.5 Reset controller

The core module FPGA incorporates a reset controller which enables the core module to be reset as a standalone unit or as part of an Integrator development system. The core module can be reset from five sources:

- reset button
- motherboard
- other core modules
- Multi-ICE
- software.

Figure 3-3 shows the architecture of the reset controller.

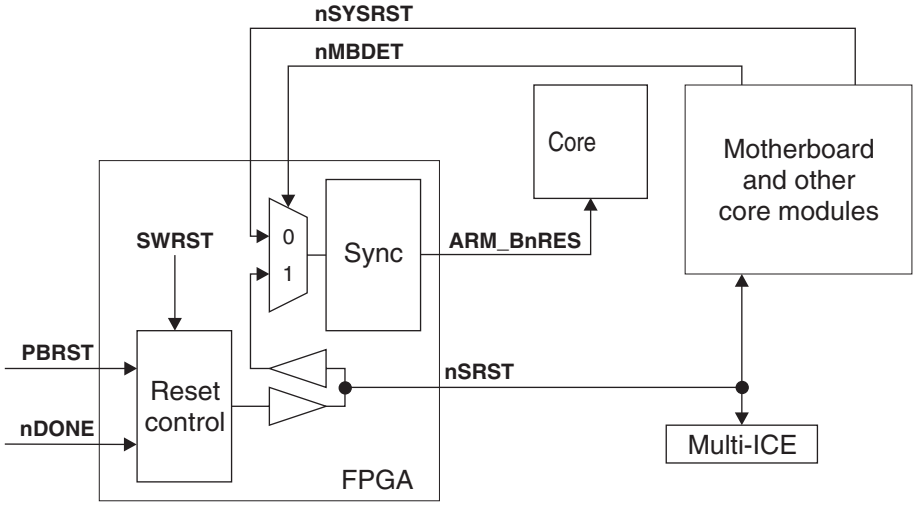


Figure 3-3 Core module reset controller

Copyright © ARM Limited 2001. All rights reserved.

3.5.1 Reset signals

Table 3-2 describes the external reset signals.

Table 3-2 Reset signal descriptions

Name	Description	Type	Function
ARM_BnRES	Processor reset	Output	The **ARM_BnRES** signal is used to reset the processor core. It is generated from **nSRST** LOW when the core module is used standalone, or **nSYSRST** LOW when the core module is attached to a motherboard. It is asserted as soon as the appropriate input becomes active. It is deasserted synchronously from the falling edge of the processor bus clock.
nDONE	FPGA configured	Input	The **nDONE** signal is an inversion of the open collector signal **FPGADONE** which is generated by all FPGAs when they have completed their configuration. The **FPGADONE** signal is routed round the system through the HDRB connectors to the inputs of all other FPGAs in the system. The signal **nSRST** is held asserted until **nDONE** is driven LOW.
nMBDET	Motherboard detect	Input	The **nMBDET** signal is pulled LOW when the core module is attached to a motherboard and HIGH when the core module is used standalone. When **MBDET** is LOW, **nSYSRST** is used to generate the **ARM_BnRES** signal. When **nMBDET** is HIGH, **nSRST** is used to generate the **ARM_BnRES** signal.
PBRST	Push-button reset	Input	The **PBRST** signal is generated by pressing the reset button.
nSRST	System reset	Bidirectional	The **nSRST** open collector output signal is driven LOW by the core module FPGA when the signal **PBRST** or software reset (**SWRST**) is asserted. As an input, **nSRST** can be driven LOW by Multi-ICE. If there is no motherboard present, the **nSRST** signal is synchronized to the processor bus clock to generate the **BnRES-M** signal.
nSYSRST	System reset	Input	The **nSYSRST** signal is generated by the system controller FPGA on the motherboard. It is used to generate the **ARM_BnRES** signal when the core module is attached to a motherboard. It is selected by the motherboard detect signal (**nMBDET**).

Copyright © ARM Limited 2001. All rights reserved.

3.5.2 Software resets

The core module FPGA provides a software reset which can be triggered by writing to the reset bit in the CM_CTRL register. This generates the internal reset signal **SWRST** that generates **nSRST** and resets the whole system (see *Core module control register* on page 4-13).

Copyright © ARM Limited 2001. All rights reserved.

3.6 System bus bridge

The system bus bridge provides an asynchronous bus interface between the local memory bus and system bus connecting the motherboard and other modules. This section describes the following details of the system bus bridge:

- *System bus bridge FIFOs* on page 3-11
- *Processor accesses to the system bus* on page 3-11
- *Motherboard accesses to SDRAM* on page 3-14
- *Multiprocessor support* on page 3-15
- *System bus signal routing* on page 3-16
- *Bus operating modes* on page 3-18.

3.6.1 System bus bridge FIFOs

Inter-module accesses are supported by two 16 x 74-bit FIFOs. Each of the 16 entries in the FIFOs contains:

- 32-bit data used for write transfers
- 32-bit address used for reads and writes
- 10-bit transaction control used for reads and writes.

3.6.2 Processor accesses to the system bus

The first FIFO supports read and write accesses by the local processor to the system bus, which extends onto the motherboard and other modules.

Processor writes

The data routing for processor writes to the system bus is illustrated in Figure 3-4 on page 3-12.

Write transactions from the processor to the system bus normally complete on the local memory bus in a single cycle. The data, address, and control information associated with the transfer are posted into FIFO, and the transfer on the system bus occurs some time later when that bus is available. This means that system bus error responses to write transfers are not reported back to the processor as data aborts. If the FIFO is full, the processor receives a wait response until space becomes available.

Copyright © ARM Limited 2001. All rights reserved.

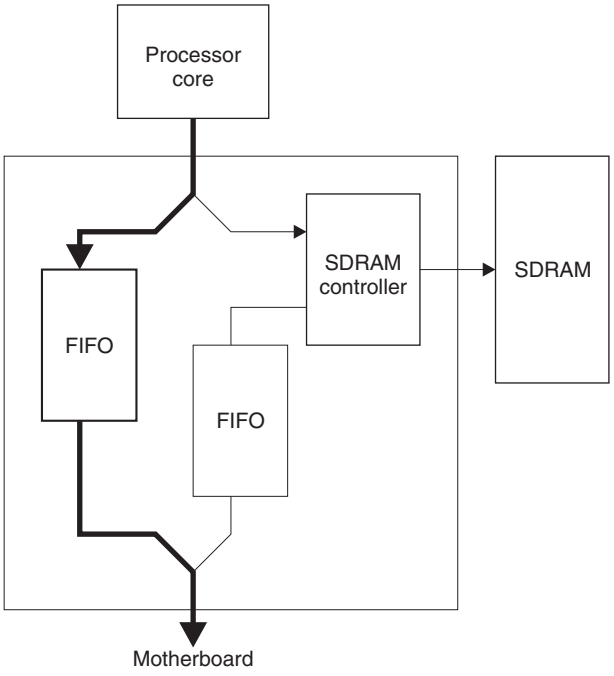


Figure 3-4 Processor writes to the system bus

Copyright © ARM Limited 2001. All rights reserved.

Processor reads

The data routing for processor reads from the system bus is illustrated in Figure 3-5.

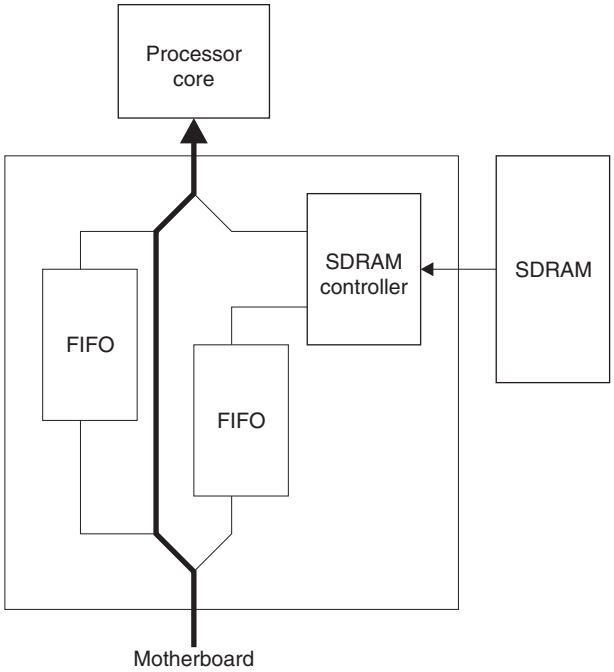


Figure 3-5 Processor reads from the system bus

For reads from the system bus, the address and control information also pass through the FIFO. The returned data from the system bus bypasses the FIFO.

The order of processor transactions is preserved on the system bus. Any previously posted writes are drained from the FIFO (that is, completed on the system bus) before the read transfer is performed. The processor receives a wait response until the read transfer has completed on the system bus, when it receives the data and any associated bus error response from the system bus. For information about SDRAM addresses, see *SDRAM mapping* on page 4-5.

You can use a system bus read to drain the FIFO of writes that would affect a subsequent operation. For example, to ensure that a write by an IRQ or FIQ handler clears an interrupt source, preventing the same interrupt being taken again by the core, the handler would follow a write to the interrupt source with a read from the system bus.

Copyright © ARM Limited 2001. All rights reserved.

3.6.3 Motherboard accesses to SDRAM

The second FIFO supports read and write accesses by system bus masters on the motherboard and other core modules to the local core module memory.

System bus writes

The data routing for system bus writes to SDRAM is illustrated in Figure 3-6.

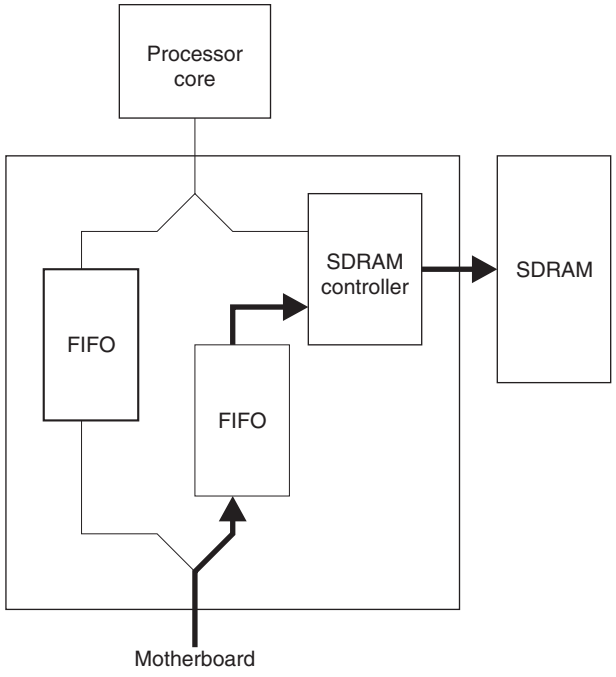


Figure 3-6 System bus writes to SDRAM

Write transactions from the system bus to the SDRAM normally complete in a single cycle on the system bus. The data, address, and control information associated with the transfer are posted into the FIFO, and the transfer into the SDRAM completes when the SDRAM is available. If the FIFO is full, then the system bus master receives a retract response indicating that the arbiter may grant the bus to another master and that this transaction must be retried later.

Copyright © ARM Limited 2001. All rights reserved.

System bus reads

The data routing for system bus reads from SDRAM is illustrated in Figure 3-7.

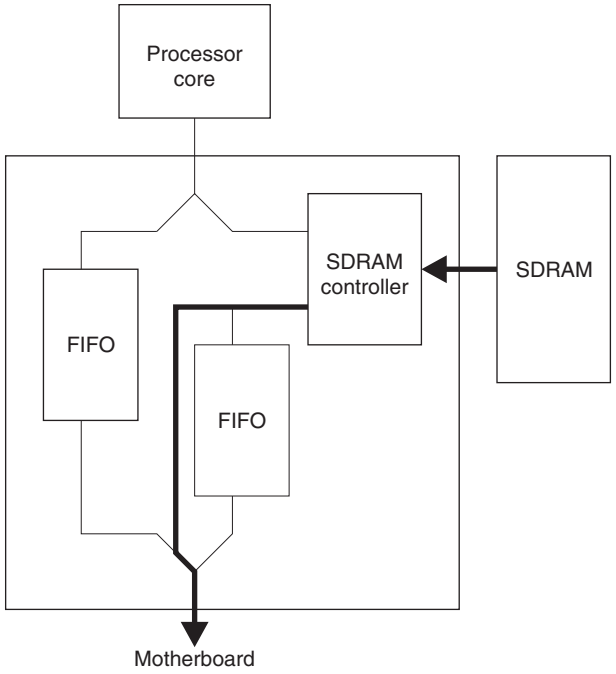


Figure 3-7 System bus reads from SDRAM

For system bus reads, the address and control information also pass through the FIFO, but the returned data from the SDRAM bypasses the FIFO.

The order of transactions on the system bus and the memory bus is preserved. Any previously posted write transactions are drained from the FIFO (that is, writes to SDRAM are completed) before the read transfer is performed.

3.6.4 Multiprocessor support

The two FIFOs operate independently, as described above, and can be accessed at the same time. This makes it possible for a local processor to read local SDRAM over the system bus (through both FIFOs). This feature can be used to support multiprocessor systems that share data in SDRAM because the processors can all access the same DRAM locations at the same addresses.

Copyright © ARM Limited 2001. All rights reserved.

3.6.5 System bus signal routing

The core module is mounted onto a motherboard using the connectors HDRA and HDRB. As well as carrying all signal connections between the boards, these provide mechanical mounting (see *Attaching the core module to a motherboard* on page 2-5).

HDRA

The signals on the HDRA connectors are tracked between the socket on the underside and the plug on the top so that all pins on the socket connect the corresponding pins on the plug. For example pin 1 connects to pin 1.

HDRB

A number of signals on the HDRB connectors are assigned to specific modules. These are rotated in groups of four between the connectors on the bottom and top of each module to ensure that each processor (or other bus master device) on a module connects to a specific signal, according the ID of the module. The ID for a module is determined by the position of the module in the stack, see *Module ID selection* on page 3-19. This signal rotation scheme is illustrated in Figure 3-8 on page 3-17.

Copyright © ARM Limited 2001. All rights reserved.

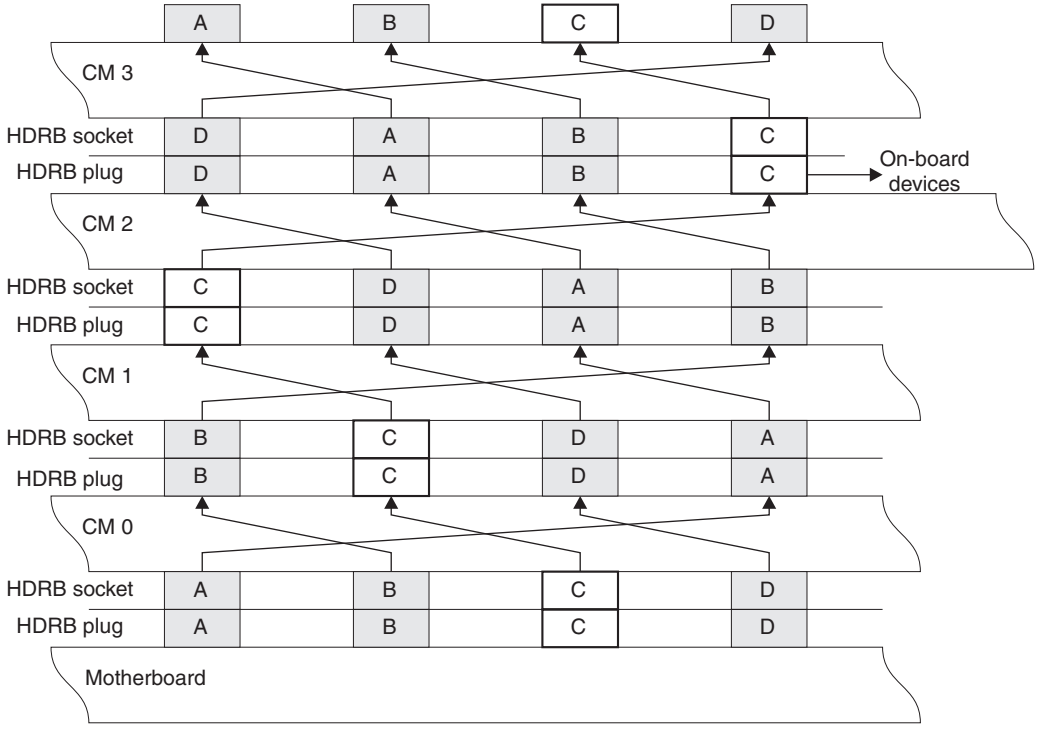


Figure 3-8 Signal rotation on HDRB

The example in Figure 3-8 on page 3-17 illustrates how a group of four signals (labeled A, B, C, and D) are routed through a group of four connector pins up through the stack. It highlights how signal C is rotated as it passes up through the stack and only utilized on module 2.

All four signals are rotated and utilized in a similar way, as follows:

- signal A is only used on core module 0
- signal B is only used on core module 1
- signal C is only used on core module 2
- signal D is only used on core module 3.

For details of the signals on the HDRB connectors, see *HDRB* on page A-4.

Note

The JTAG signals are discussed in *Multi-ICE support* on page 3-26.

Copyright © ARM Limited 2001. All rights reserved.

3.6.6 Bus operating modes

The bus operating modes are programmed by writing to coprocessor 15 register 1 within the microprocessor core.

The Integrator/CM920T-ETM supports:

- asynchronous and FastBus clocking
- little-endian addressing.

The Integrator/CM920T-ETM does not support:

- synchronous clocking
- big-endian addressing.

For details of how to set the bus operating parameters, refer to the *ARM920T Technical Reference Manual*.

Copyright © ARM Limited 2001. All rights reserved.

3.7 Module ID selection

The position of a core module in the HDRA/HDRB stack is used to determine its ID, and from this its address in the alias memory region (see the user guide for your motherboard) and the interrupts that it responds to.

Note

The core module cannot be damaged by connecting it onto the EXPA/EXPB position on the Integrator/AP motherboard, but fitting it in this position prevents correct operation.

3.7.1 Module address decoding

The Integrator system implements a distributed address decoding system. These means that each core or logic module must decode its own area of the memory map. The central decoder in the system controller FPGA (on the motherboard) responds with an error response for all areas of the address space that are not occupied by a module. This default response is disabled for a memory region occupied by a module that is fitted.

The signals **nPPRES[3:0]** (core module present) **nEPRES[3:0]** (logic module present) are used to signal the presence of modules to the central decoder. The signals **ID[3:0]** indicate to the module its position in the stack and the address range that its own decoder must respond to. These signals rotate as they pass up the stack, as described in *System bus signal routing* on page 3-16. Only one signal in each group is pulled LOW for each module.

The alias SDRAM address of a core module is determined in hardware, although a module can determine its own position by reading a decoded version of **ID[3:0]** from the CM_STAT register (see *Core module status register* on page 4-14). Table 3-3 shows alias addresses for a core module fitted to a the motherboard on the HDRA/HDRB stack.

Table 3-3 Core module address decode

ID[3:0]	Module ID	Address range	Size
1101	3 (top)	0xB0000000 to 0xBFFFFFFF	256MB
1011	2	0xA0000000 to 0xAFFFFFFF	256MB
0111	1	0x90000000 to 0x9FFFFFFF	256MB
1110	0 (bottom)	0x80000000 to 0x8FFFFFFF	256MB

Copyright © ARM Limited 2001. All rights reserved.

3.7.2 Interrupts

The system controller FPGA on the motherboard incorporates an interrupt controller that routes the various interrupts from around the system onto the **nFIQ** and **nIRQ** pins of up to four processors. The interrupts that a core module receives are determined by the position of the core module within the stack, as shown in Table 3-4.

Table 3-4 Core module interrupts

Module ID	Interrupt	Fast interrupt
3 (top)	**nIRQ3**	**nFIQ3**
2	**nIRQ2**	**nFIQ2**
1	**nIRQ1**	**nFIQ1**
0 (bottom)	**nIRQ0**	**nFIQ0**

The interrupt signals are routed to the core module using pins on the HDRB connectors (see *HDRB* on page A-4).

The interrupts and fast interrupts are enabled and handled using the interrupt control registers on the motherboard (see the user guide for your motherboard).

Copyright © ARM Limited 2001. All rights reserved.

3.8 Clock generators

The core module provides its own clock generators and operates asynchronously with the motherboard. The clock generators provide three programmable clock sources. In addition, a fixed-frequency reference clock **REFCLK** is supplied to the FPGA. These are discussed in the following subsections:

- *Clock generation functional overview* on page 3-21
- *Programming the processor core clock (CORECLK)* on page 3-22
- *Programming the local memory bus clock (LCLK)* on page 3-23.
- *Programming the auxiliary clock (AUXCLK)* on page 3-24
- *FPGA reference clock (REFCLK)* on page 3-25.

3.8.1 Clock generation functional overview

The architecture of the clock generators is shown in Figure 3-9.

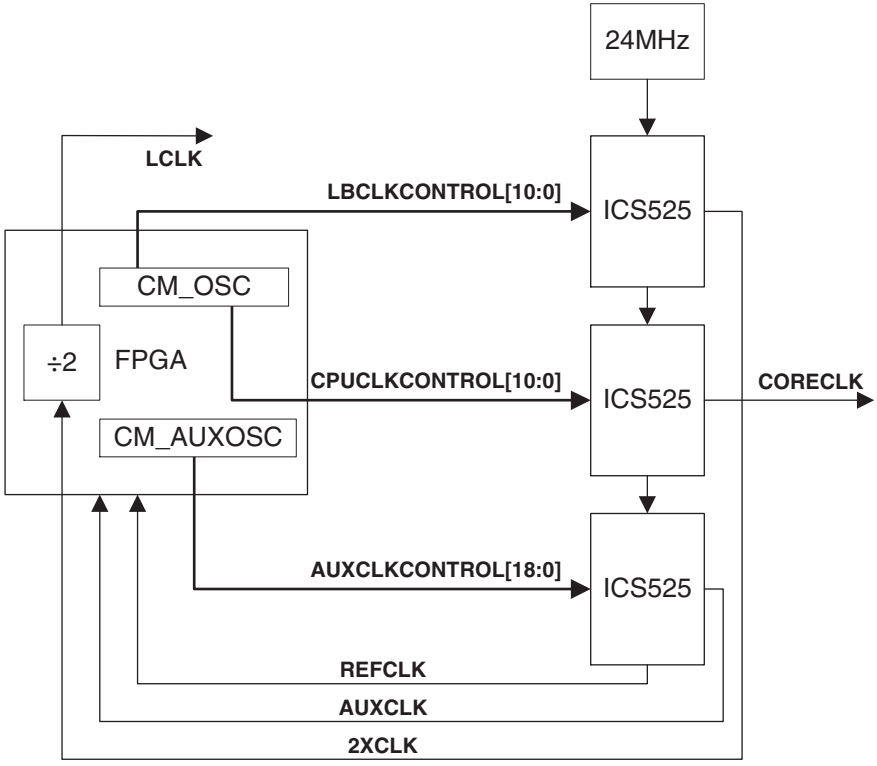


Figure 3-9 Core module clock generator

Copyright © ARM Limited 2001. All rights reserved.

The core module has three Microclock OSCaR programmable clock generators. These are supplied with a reference clock by a 24MHz crystal oscillator and their output frequencies are controlled by *divider* input pins. They are able to produce a wide range of frequencies controlled by registers in the FPGA.

3.8.2 Programming the processor core clock (CORECLK)

The frequency of **CORECLK** is controllable in 1MHz steps in the range 12MHz to 160MHz. This is achieved by setting the *Voltage Controlled Oscillator* (VCO) divider and output divider for the **CORECLK** generator in the CM_OSC register. The VCO divider is controlled by the C_VDW bits and output divider is controlled by the C_OD bits. The reference divider value is fixed.

Figure 3-10 shows the values placed on the divider input pins and how the clock speeds are obtained.

C_RDW R[6:0]							C_VDW V[9:0]									C_OD S[2:0]		
0	0	1	0	1	1	0	0	C	C	C	C	C	C	C	C	C	C	C
22 (fixed value)							4 to 152									2 to 10		

Figure 3-10 CORECLK divider control

The bits marked:

- C are programmable in the CM_OSC register
- 1 are tied HIGH
- 0 are tied LOW.

The frequency of **CORECLK** can be calculated using the formula:

freq = 2*((C_VDW+ 8)/C_OD)

where:

C_VDW is the VCO divider word for the core clock.

C_OD is the output divider for the core clock.

Note

The output divider is not a straight forward binary representation of a decimal number. Its value is assigned as described in *Programming the output dividers* on page 3-25.

For details about programming the CM_OSC register, see *Core module oscillator register* on page 4-12.

Copyright © ARM Limited 2001. All rights reserved.

Note

Values for C_VDW and C_OD can be calculated using the ICS525 calculator on the Microclock website.

The **CORECLK** is converted to testchip pads supply level by the FPGA. The clock is series terminated with a 33Ω resistor and then drives a single load on the microprocessor core.

3.8.3 Programming the local memory bus clock (LCLK)

The clock signal **2XCLK** is divided by 2 by the FPGA to produce **LCLK** which is distributed to the following devices:

- FPGA
- Logic analyzer
- SDRAM DIMM
- SSRAM PLD
- ARM920T test chip

Note

The clock is fed back into the FPGA so that the internal version is not skewed with respect to the others.

The frequency of **LCLK** is controllable in 0.5MHz steps in the range 6MHz to 66MHz by programming the VCO and output divider bits for the **2XCLK** generator in the CM_OSC register. The VCO divider is controlled by the L_VDW bits and the output divider is controlled by the L_OD bits. The reference divider is fixed. The default frequency for the core module is 40MHz. Figure 3-11 shows the values placed on the divider input pins and how the clock speeds are obtained.

L_RDW R[6:0]							L_VDW V[9:0]									L_OD S[2:0]		
0	0	1	0	1	1	0	0	L	L	L	L	L	L	L	L	L	L	L
22 (fixed value)							4 to 124									2 to 10		

Figure 3-11 2XCLK divider control

The bits marked:

- L are programmable in the CM_OSC register
- 1 are tied HIGH
- 0 are tied LOW.

Copyright © ARM Limited 2001. All rights reserved.

The frequency of the **LCLK** can be calculated using the simplified formula:

freq = (L_VDW+ 8)/L_OD

where:

L_VDW is the VCO divider word.

L_OD is the output divider.

Note

The output divider is not a straight forward binary representation of a decimal number. Its value is assigned as described in *Programming the output dividers* on page 3-25.

For details about programming L_VDW and L_OD, see *Core module oscillator register* on page 4-12.

3.8.4 Programming the auxiliary clock (AUXCLK)

The frequency of the **AUXCLK** is controlled by the register CM_AUXOSC (see *Core module auxiliary oscillator register* on page 4-17). The values for RDW, VDW, and OD are all programmable, as shown in Figure 3-12, to give frequencies in the range 1 to 160MHz with a better than 0.1% accuracy. The default frequency following a power-on reset is 32.369MHz.

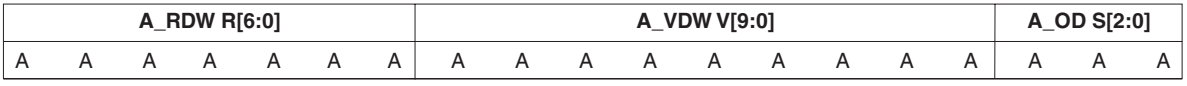


Figure 3-12 AUXCLK divider control

The clock frequency is controlled by the divider signals **AUXCLKCONTROL[18:0]** that are assigned to the control parameters as shown in Table 3-5 on page 3-25.

Copyright © ARM Limited 2001. All rights reserved.

Table 3-5 Clock control signal assignment

Signals	Control parameter	Label
AUXCLKCONTROL[18:16]	Output divider	A_OD
AUXCLKCONTROL[15:9]	Reference divider	A_RDW
AUXCLKCONTROL[8:0]	VCO divider	A_VDW

The reference divider and VCO divider are used to calculate the output frequency using the following formula:

freq = 48*(A_VDW+ 8)/((A_RDW+2)*A_OD)

3.8.5 Programming the output dividers

The output dividers, C_OD, L_OD, and A_OD, are not a straight forward binary representations of a decimal values but have assigned values. The value assigned to each bit pattern is shown in Table 3-6 on page 3-25, where S is

Table 3-6 Values for output divider

OD[2:0]	Value
001	2
011	4
100	5
111	6
101	7
010	8
110	9
000	10

3.8.6 FPGA reference clock (REFCLK)

The **REFCLK** signal is used by the FPGA to generate the SDRAM refresh clock and SPD EEPROM clock.

Copyright © ARM Limited 2001. All rights reserved.

3.9 Multi-ICE support

The core module supports debugging using JTAG. This is described in the following subsections:

- *Multi-ICE connection* on page 3-26
- *JTAG scan paths* on page 3-27
- *JTAG connection modes* on page 3-29
- *JTAG signals* on page 3-30.

3.9.1 Multi-ICE connection

Figure 3-13 shows the Multi-ICE connector and the CONFIG link.

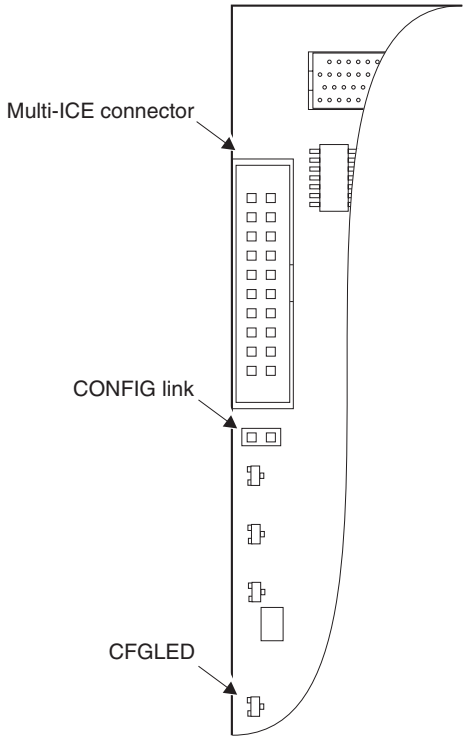


Figure 3-13 JTAG connector, CONFIG link, and LED

The CONFIG link is used to enable in-circuit programming of the FPGA and PLDs using Multi-ICE (see *JTAG connection modes* on page 3-29).

Copyright © ARM Limited 2001. All rights reserved.

The Multi-ICE connector provides a set of JTAG signals that allow JTAG debugging equipment to be used (see *JTAG signals* on page 3-30). If you are debugging a development system with multiple core modules, connect the JTAG debugging equipment to the top core module.

3.9.2 JTAG scan paths

This section describes JTAG scan chain data path and clock path.

Data path

Figure 3-14 shows a simplified diagram of the data path.

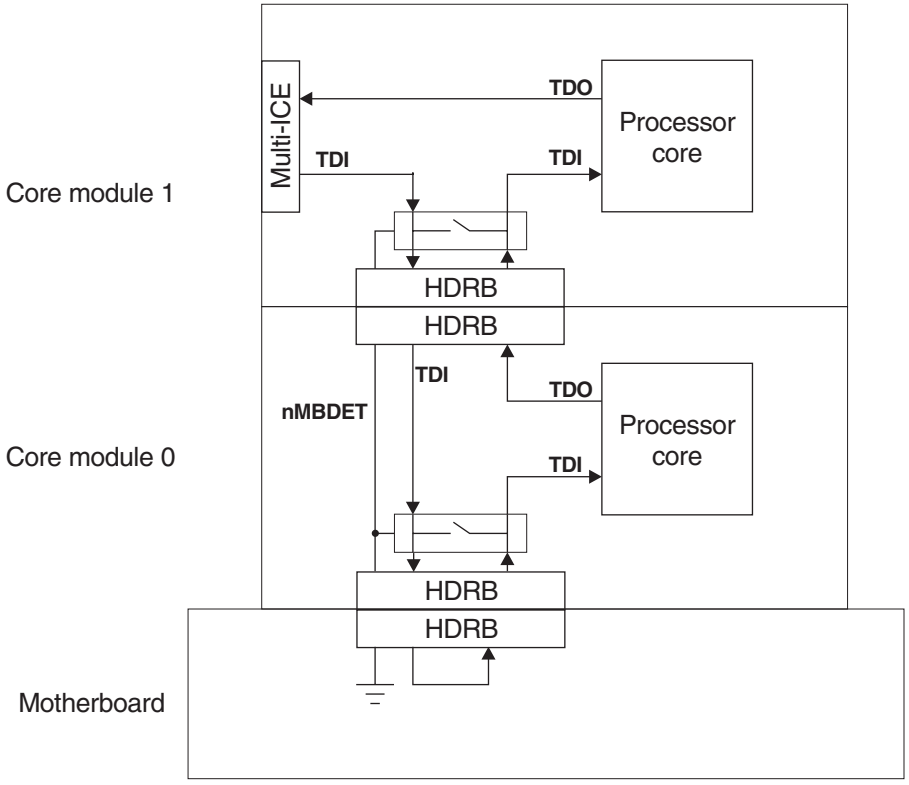


Figure 3-14 JTAG data path

When you use the core module as a standalone development system, the data path is routed to the processor core and back to the Multi-ICE connector.

Copyright © ARM Limited 2001. All rights reserved.

If the core module is attached to an Integrator motherboard, the **TDI** signal from the top core module is routed down through the HDRB connectors of any modules in the stack to the motherboard. From there the path is routed back up the stack through each core module, before being returned to the Multi-ICE connector as **TDO**. The motherboard detect signal **nMBDET** controls a switching circuit on the core module and, therefore, the routing of **TDI**.

The PLDs and FPGAs are included in the scan chain if the core module is in configuration mode, as described in *JTAG connection modes* on page 3-29.

Clock path

The clock path is routed in a similar way to the data path, although in the opposite direction. Figure 3-15 shows a simplified diagram of the clock path.

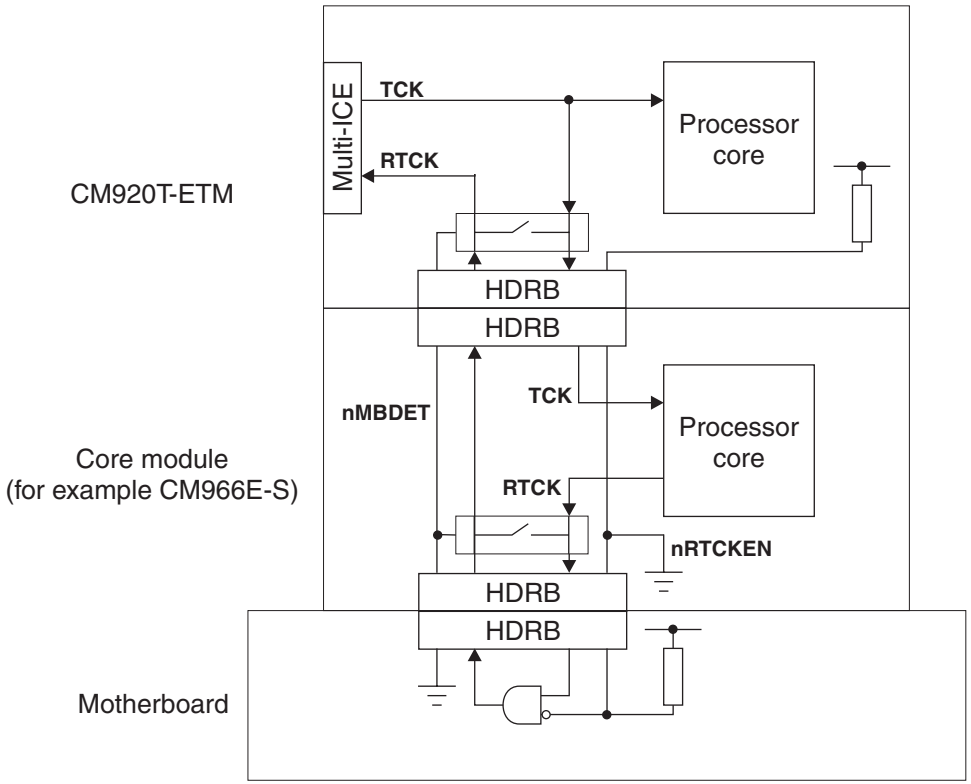


Figure 3-15 JTAG clock path

Copyright © ARM Limited 2001. All rights reserved.

A number of synthesized cores (for example, the ARM966E-S) sample **TCK**. This introduces a delay into the clock path. Cores of this type pass on the delayed clock signal as **RTCK**, which is fed to the **TCK** input of the next device in the chain. The **RTCK** signal at the Multi-ICE connector is used by Multi-ICE to regulate the advance of **TCK**, a mechanism called adaptive clocking (see the *ARM Multi-ICE User Guide*).

The routing of the **TCK/RTCK** signals through the stack is controlled by switches in a similar way to the data path. The routing of **RTCK** back up the stack is controlled by the signal **nRTCKEN** and an AND gate on the motherboard (the pullups on **nMBDET** are omitted for clarity).

The ARM920T does not sample **TCK** but routes the **TCK** signal straight through to the next board down the stack. If one or more modules in a stack drives **RTCK** (and so asserts **nRTCKEN**), you must ensure that the board at the bottom of the stack provides the necessary return path. All Integrator motherboards do so.

3.9.3 JTAG connection modes

The core module is capable of operating in normal debug mode or configuration mode.

Normal debug mode

During normal operation and software development, the core module operates in debug mode. The debug mode is selected by default (when a jumper is *not* fitted at the CONFIG link, see Figure 3-13 on page 3-26). In this mode, the processor core and debuggable devices on other modules are accessible on the scan chain, as shown in Figure 3-14 on page 3-27.

Configuration mode

In configuration mode the debuggable devices are still accessible and, in addition, all FPGAs and PLDs in the system are added into the scan chain. This allows the board to be configured or upgraded in the field using Multi-ICE or other JTAG debugging equipment.

To select configuration mode, fit a jumper to the CONFIG link on the core module *at the top of the stack (*see Figure 3-13 on page 3-26*)*. This has the effect of pulling the **nCFGEN** signal LOW which illuminates the CFG LED (yellow) on each module in the stack and reroutes the JTAG scan path. The LED provides a warning that the development system is in the configuration mode.

Copyright © ARM Limited 2001. All rights reserved.

Note

Configuration mode is guaranteed for a single core module attached to a motherboard but may be unreliable if more than one core module is attached. The larger loads on the **TCK** and **TMS** lines could cause unreliable operation.

After configuration or code updates you must:

1. Remove the CONFIG link.
2. Power cycle the development system.

The configuration mode allows FPGA and PLD code to be updated as follows:

- The FPGAs are volatile, but load their configuration from flash memory. Flash memory, which itself does not have a JTAG port, can be programmed by loading designs into the FPGAs and PLDs which handle the transfer of data to the flash using JTAG.
- The PLDs are nonvolatile devices which can be programmed directly by JTAG.

3.9.4 JTAG signals

Figure 3-16 shows the pinout of the Multi-ICE connector and Table 3-7 on page 3-32 provides a description of the JTAG and related signals.

Note

In the description in Table 3-7 on page 3-32, the term JTAG equipment refers to any hardware that can drive the JTAG signals to devices in the scan chain.Typically this will be Multi-ICE, although hardware from other suppliers can also be used to debug ARM processors.

 Copyright © ARM Limited 2001. All rights reserved.

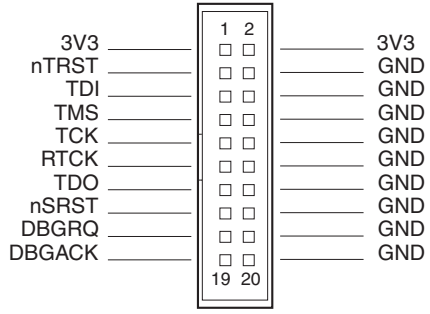


Figure 3-16 Multi-ICE connector pinout

Copyright © ARM Limited 2001. All rights reserved.

Table 3-7 JTAG signal description

Name	Description	Function
DBGRQ	Debug request (from JTAG equipment)	**DBGRQ** is a request for the processor core to enter the debug state. It is provided for compatibility with third-party JTAG equipment.
DBGACK	Debug acknowledge (to JTAG equipment)	**DBGACK** indicates to the debugger that the processor core has entered debug mode. It is provided for compatibility with third-party JTAG equipment.
DONE	FPGA configured	**DONE** is an open-collector signal which indicates when FPGA configuration is complete. Although this signal is not a JTAG signal, it does effect **nSRST**. The **DONE** signal is routed between all FPGAs in the system through the HDRB connectors. The master reset controller on the motherboard senses this signal and holds all the boards in reset (by driving **nSRST** LOW) until all FPGAs are configured.
nCFGEN	Configuration enable (from jumper on module at the top of the stack)	**nCFGEN** is an active LOW signal used to put the boards into configuration mode. The **nCFGEN** signal is routed between all FPGAs in the system through the HDRB connectors. In configuration mode all FPGAs and PLDs are connected to the scan chain so that they can be configured by the JTAG equipment.
nRTCKEN	Return TCK enable (from core module to motherboard)	**nRTCKEN** is an active LOW signal driven by any core module that requires **RTCK** to be routed back to the JTAG equipment. If **nRTCKEN** is HIGH, the motherboard drives **RTCK** LOW. If **nRTCKEN** is LOW, the motherboard drives the **TCK** signal back up the stack to the JTAG equipment. The **nCFGEN** signal is routed between all FPGAs in the system through the HDRB connectors.
nSRST	System reset (bidirectional)	**nSRST** is an active LOW open-collector signal which can be driven by the JTAG equipment to reset the target board. Some JTAG equipment senses this line to determine when a board has been reset by the user. When the signal is driven LOW by the reset controller on the core module, the motherboard resets the whole system by driving **nSYSRST** LOW. This is also used in configuration mode to control the initialization pin (**nINIT**) on the FPGAs. Though not a JTAG signal, **nSRST** is described because it can be controlled by JTAG equipment.

Copyright © ARM Limited 2001. All rights reserved.

Table 3-7 JTAG signal description (continued)

Name	Description	Function
nTRST	Test reset (from JTAG equipment)	This active low open-collector is used to reset the JTAG port and the associated debug circuitry on the ARM920T processor. It is asserted at power-up by each module, and can be driven by the JTAG equipment. This signal is also used in configuration mode to control the programming pin (**nPROG**) on FPGAs.
RTCK	Return TCK (to JTAG equipment)	Some devices sample **TCK** (for example a synthesizable core with only one clock), and this has the effect of delaying the time at which a component actually captures data. Using a mechanism called *adaptive clocking* , the **RTCK** signal is returned by the core to the JTAG equipment, and the clock is not advanced until the core has captured the data. In *adaptive clocking mode*, Multi-ICE waits for an edge on **RTCK** before changing **TCK**. In a multiple device JTAG chain, the **RTCK** output from a component connects to the **TCK** input of the next device in the chain. The **RTCK** signal on the module connectors HDRB returns **TCK** to the JTAG equipment. If there are no synchronizing components in the scan chain then it is unnecessary to use the **RTCK** signal and it is connected to ground on the motherboard.
TCK	Test clock (from JTAG equipment)	**TCK** synchronizes all JTAG transactions. **TCK** connects to all JTAG components in the scan chain. Series termination resistors are used to reduce reflections and maintain good signal integrity. **TCK** flows down the stack of modules and connects to each JTAG component. However, if there is a device in the scan chain that synchronizes **TCK** to some other clock, then all down-stream devices are connected to the **RTCK** signal on that component (see **RTCK**).
TDI	Test data in (from JTAG equipment)	**TDI** goes down the stack of modules to the motherboard and then back up the stack, labelled **TDO**, connecting to each component in the scan chain.
TDO	Test data out (to JTAG equipment)	**TDO** is the return path of the data input signal **TDI**. The module connectors HDRB have two pins labelled **TDI** and **TDO**. **TDI** refers to data flowing down the stack and **TDO** to data flowing up the stack. The JTAG components are connected in the return path so that the length of track driven by the last component in the chain is kept as short as possible.
TMS	Test mode select (from JTAG equipment)	**TMS** controls transitions in the tap controller state machine. **TMS** connects to all JTAG components in the scan chain as the signal flows down the module stack.

Copyright © ARM Limited 2001. All rights reserved.

3.9.5 Debug communications interrupts

The ARM920T processor core incorporates EmbeddedICE logic and provides a debug communications data register that is used to pass data between the processor and JTAG equipment. The processor accesses this register as a normal 32-bit read/write register and the JTAG equipment reads and writes the register using a scan chain. For a description of the debug communications channel, see the *ARM920T Technical Reference Manual*.

Interrupts can be used to signal when data has been written into one side of the register and is available for reading from the other side. These interrupts are supported by the interrupt controller within the core module FPGA and can be enabled and cleared by accessing the interrupt registers (see *Core module interrupt registers* on page 4-23).

 Copyright © ARM Limited 2001. All rights reserved.

3.10 Embedded Trace support

The ARM920T-ETM processors incorporate an *ARM9 Embedded Trace Macrocell* (ETM9). This enables you to carry out real-time debugging by connecting external trace equipment to the core module. To trace program flow, the ETM broadcasts branch addresses, data accesses, and status information through the trace port. Later in the debug process, the complete instruction flow can be reconstructed by the ARM *Trace Debug Tools* (TDT).

Note

It is not possible to reconstruct self-modifying code.

3.10.1 About using trace

Figure 3-17 illustrates a trace debugging setup with the core module.

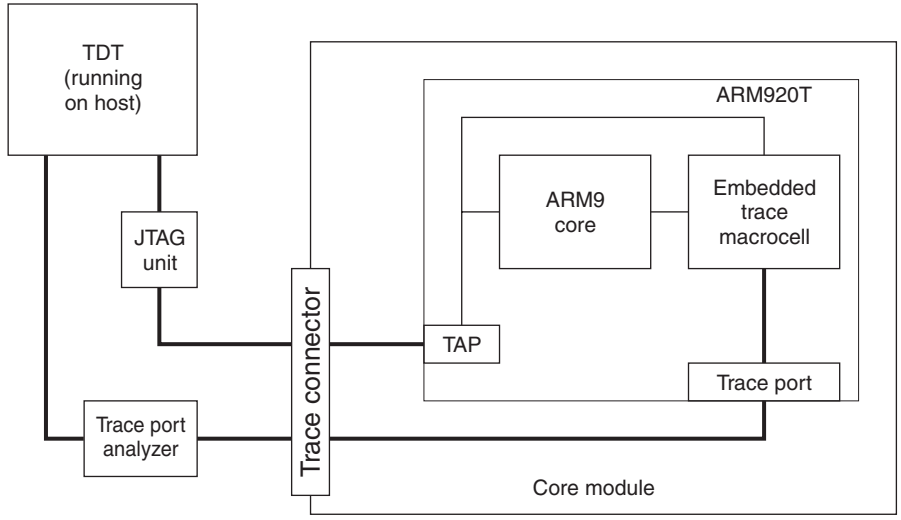


Figure 3-17 Trace connection

Note

The routing of the JTAG scan chain on the Integrator system is described in *Multi-ICE support* on page 3-26.

Copyright © ARM Limited 2001. All rights reserved.

The components in the trace debug setup shown in Figure 3-17 are as follows:

Embedded trace macrocell

The ETM monitors the ARM core buses and outputs compressed information through the trace port to a trace port analyzer. The on-chip ETM contains trigger and filter logic to control what is traced.

Trace port analyzer

The *Trace Port Analyzer* (TPA) is an external device that stores information from the trace port.

JTAG unit

This is a protocol converter that converts debug commands from the debugger into JTAG messages for the ETM. The JTAG unit can be a separate device, such as Multi-ICE, or can be incorporated within the TPA.

Trace debug tools

The *Trace Debug Tools* (TDT) is an optional component of the *ARM Developer Suite* (ADS) that runs on a host system. It is used to set up the filter logic, retrieve data from the analyzer, and reconstruct an historical view of processor activity. For further information, see the *ADS Trace Debug Tools User Guide*.

3.10.2 Core trace configuration

The ETM configuration provided by the test chip is defined by its manufacturer, see the release note for your test chip.

3.10.3 Trace interface description

The trace connector enables you to connect an external embedded trace interface module. This connector is a high-density AMP Mictor connector. The pinout for this connector is provided in *Trace connector pinout* on page A-10.

 Copyright © ARM Limited 2001. All rights reserved.

Chapter 4
Programmer's Reference

This chapter describes the memory map and the status and control registers. It contains the following sections:

- *Memory organization* on page 4-2
- *Exception vector mapping* on page 4-8
- *Core module registers* on page 4-9
- *Core module flag registers* on page 4-22
- *Core module interrupt registers* on page 4-23
- *SDRAM SPD memory* on page 4-27.

Copyright © ARM Limited 2001. All rights reserved.

4.1 Memory organization

This section describes the memory map of the core module. For a standalone core module, the memory map is limited to local SSRAM, SDRAM, and core module registers. The memory map depends on whether the module is fitted to a motherboard and on the state of REMAP.

For the full memory map of an Integrator development system, which includes a motherboard, refer to the user guide for the motherboard.

4.1.1 Core module memory map

The core module has a fixed memory map that maintains compatibility with ARM Integrator motherboards and modules. Table 4-1 on page 4-2 shows the memory map (An x indicates that the signal can be HIGH or LOW without effect.). The signal **nMBDET** is pulled LOW when the core module is fitted to an Integrator motherboard.

Table 4-1 Core module memory map

nMBDET	REMAP	Address range	Size	Description
0	0	`0x00000000` to `0x000FFFFF`	1MB	Boot ROM or flash on motherboard
0	1	`0x00000000` to `0x000FFFFF`	1MB	SSRAM
1	x	`0x00000000` to `0x000FFFFF`	1MB	SSRAM
x	x	`0x00100000` to `0x0FFFFFFF`	255MB	Local SDRAM
x	x	`0x10000000` to `0x107FFFFF`	8MB	Core module registers
x	x	`0x10800000` to `0x10FFFFFF`	8MB	SSRAM alias
0	x	`0x11000000` to `0xFFFFFFFF`	3824MB	System bus address space
1	x	`0x11000000` to `0xFFFFFFFF`	3824MB	Bus error response

Copyright © ARM Limited 2001. All rights reserved.

4.1.2 Using REMAP

The SSRAM on the core module and the alias of the boot ROM or flash memory on an Integrator motherboard share the same locations within the Integrator memory map. Accesses these devices are controlled by the REMAP bit and the motherboard detect signal (**nMBDET**) that is permanently grounded by the motherboard. The effect on the memory map is shown in Figure 4-1.

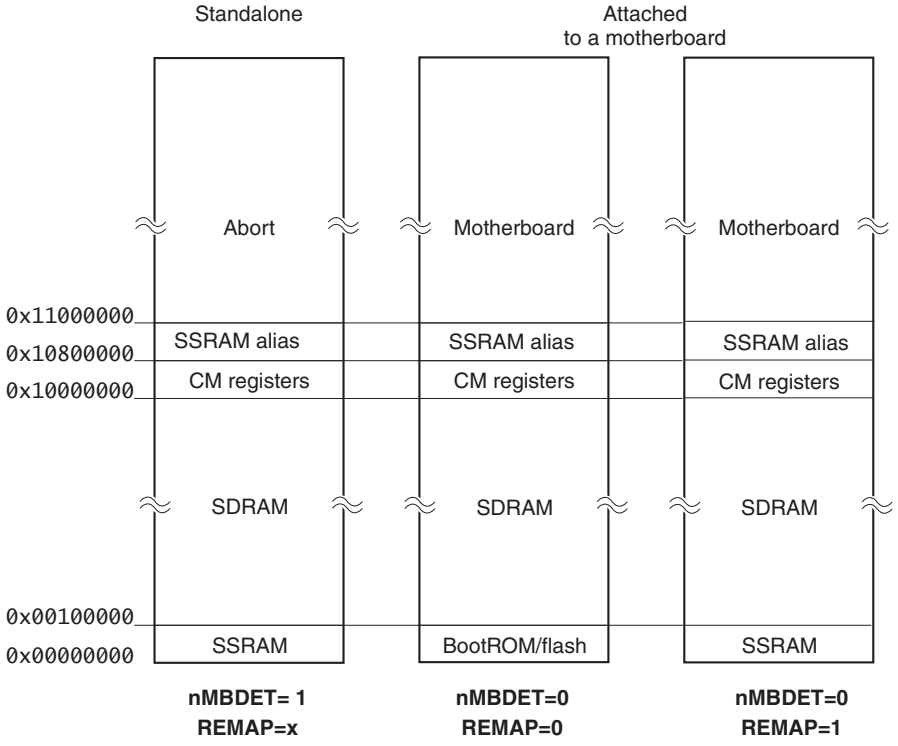


Figure 4-1 Effect of remap

The REMAP bit only has any effect if the core module is attached to a motherboard (**nMBDET**=0). It is controlled by bit 2 of the CM_CTRL register at `0x1000000C` and functions as follows:

REMAP=0 Accesses to locations within the address range `0x00000000` to `0x000FFFFF` are to the boot ROM or flash on the motherboard.

If the core module is attached to a motherboard, REMAP is always 0 following a reset.

Copyright © ARM Limited 2001. All rights reserved.

Note

You can set REMAP to 0 only if the core module is attached to a motherboard.

REMAP=1 Accesses to locations within the address range `0x00000000` to `0x000FFFFF` are to the SSRAM on the core module.

Note

Program execution normally starts at `0x00000000`. A switch on the motherboard determines which of the boot ROM or flash is aliased to this location. This enables you to boot from the boot ROM or from flash. Refer to the user guide for your motherboard for more information.

 Copyright © ARM Limited 2001. All rights reserved.

4.1.3 SDRAM mapping

The Integrator memory map provides two regions in which the SDRAM can be accessed. One region allows access only by the local processor, and the second allows access by any master within the Integrator system.

Local SDRAM access

The local SDRAM access region allows access to the SDRAM only by the processor on the same core module at `0x00100000` `0x0FFFFFFF`. You can fit an SDRAM of up to 256MB but you cannot access the lowest 1MB. This is because the lowest 1MB (`0x00000000` to `0x000FFFFF`) is hidden by the SSRAM (if REMAP=1), or by the boot ROM or flash (if REMAP=0). When a smaller sized SDRAM DIMM is fitted, it is mapped repeatedly to fill this space. Figure 4-2 shows an example of a 64MB DIMM mapped four times.

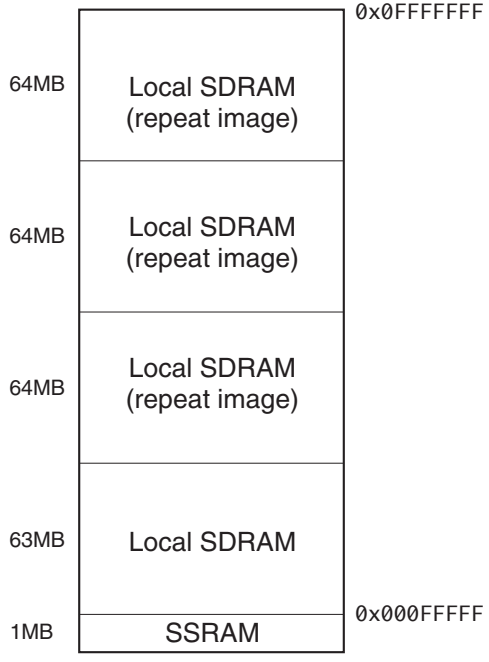


Figure 4-2 SDRAM repeat mapping for a 64MB DIMM

Copyright © ARM Limited 2001. All rights reserved.

Global SDRAM access

The Integrator system memory map provides an alias memory region in which the SDRAM on core modules can be accessed. This region is only available if the core module is attached to a motherboard. Each core module is assigned to a 256MB space within this region, allowing you to access the whole of a 256MB DIMM. The SDRAM can be accessed by all bus masters at its alias location.

The system bus address for a core module is automatically defined by its position in the stack (see *Core module ID* on page 2-6). Figure 4-3 shows the local and alias address of the SDRAM on four core modules.

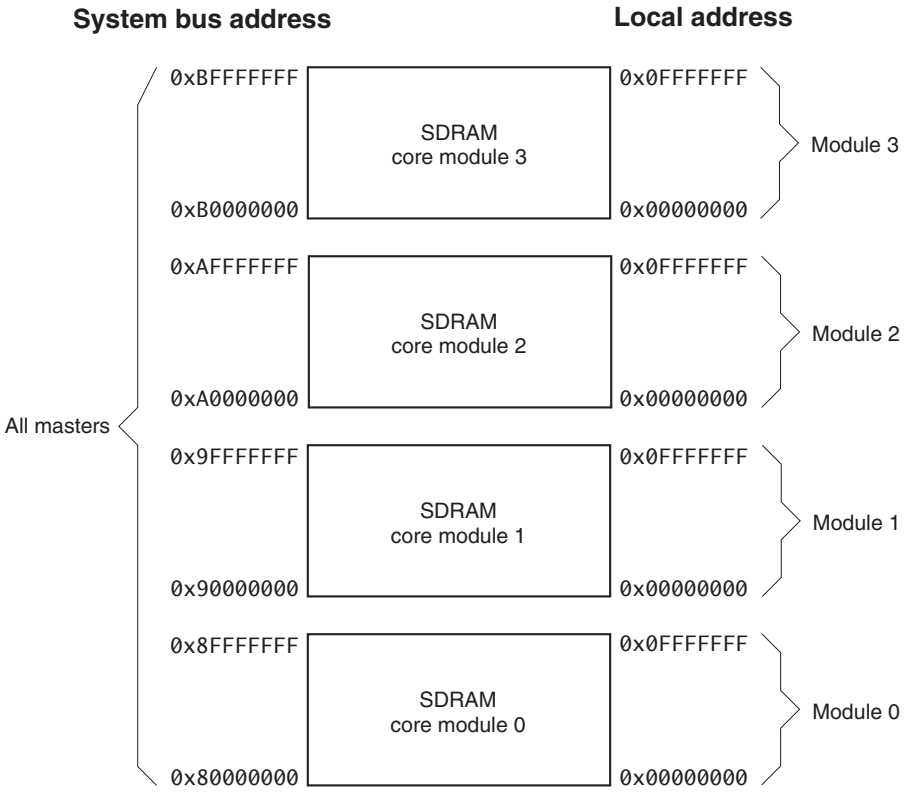


Figure 4-3 Core module local and alias addresses

Copyright © ARM Limited 2001. All rights reserved.

Note

A processor can read from and write to its own SDRAM at the system bus address. However, these accesses are slower than local accesses because they are bridged to and from the system bus.

You can determine which core module a processor is on by reading the CM_STAT register. Form this you can determine that the alias location of the SDRAM on the same core module (see *Core module status register* on page 4-14).

Copyright © ARM Limited 2001. All rights reserved.

4.2 Exception vector mapping

The convention for ARM cores is to map the exception vectors to begin at address 0. However, the ARM920T core allows the vectors to be moved to `0xFFFF0000` by setting the V bit in coprocessor 15 register 1. To maintain compatibility across all cores, the default reset value maps the vector to begin at address 0 (see the *ARM920T Technical Reference Manual*).

The CM_INIT register can be used to control the value of the V bit at reset, (see *Core module initialization register* on page 4-21).

To run applications with high vectors (at virtual address `0xFFFF0000` to `0xFFFF001C`), you must write the correct value to the coprocessor register. This virtual address should be mapped to a physical address that contains real memory, for example, core module SDRAM.

Copyright © ARM Limited 2001. All rights reserved.

4.3 Core module registers

The core module status and control registers allow the processor to determine its environment and to control some core module operations. The registers, listed in Table 4-2 on page 4-9, are located at `0x10000000` and can only be accessed by the local processor.

Table 4-2 Core module status, control, and interrupt registers

Register Name	Address	Access	Reset	Description
CM_ID	`0x10000000`	R	Static	Core module identification register
CM_PROC	`0x10000004`	R	Static	Core module processor register
CM_OSC	`0x10000008`	R/W	POR	Core module oscillator register
CM_CTRL	`0x1000000C`	R/W	Reset	Core module control
CM_STAT	`0x10000010`	R	Reset	Core module status
CM_LOCK	`0x10000014`	R/W	Reset	Core module lock
CM_LMBUSCNT	`0x10000018`	R	Reset	Core module local memory bus cycle counter
CM_AUXOSC	`0x1000001C`	R/W	POR	Core module auxiliary clock oscillator register
CM_SDRAM	`0x10000020`	R/W	POR	SDRAM status and control register
CM_INIT	`0x10000024`	R/W	POR	Core module initialization register
CM_REFCNT	`0x10000028`	R	Reset	Reference clock cycle counter
CM_UNUSED1	`0x1000002C`	-	-	Reserved
CM_FLAGS	`0x10000030`	R	Reset	Core module flag register
CM_FLAGSET	`0x10000030`	W	Reset	Core module flag set register
CM_FLAGSCLR	`0x10000034`	W	Reset	Core module flag clear register
CM_NVFLAGS	`0x10000038`	R	POR	Core module nonvolatile flag register
CM_NVFLAGSSET	`0x10000038`	W	POR	Core module nonvolatile flag set register
CM_NVFLAGSCLR	`0x1000003C`	W	POR	Core module nonvolatile flag clear register
CM_IRQ_STATUS	`0x10000040`	R	Reset	Core module IRQ status register
CM_IRQ_RSTAT	`0x10000044`	R	Reset	Core module IRQ raw status register

Copyright © ARM Limited 2001. All rights reserved.

Table 4-2 Core module status, control, and interrupt registers

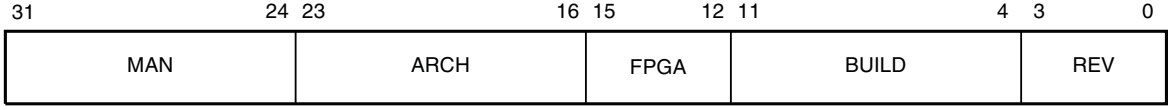
Register Name	Address	Access	Reset	Description
CM_IRQ_ENSET	0x10000048	R	Reset	Core module IRQ enable set register
CM_IRQ_ENCLR	0x1000004C	W	Reset	Core module IRQ enable clear register
CM_SOFT_INTSET	0x10000050	R/W	Reset	Core module software interrupt set
CM_SOFT_INTCLR	0x10000054	W	Reset	Core module software interrupt clear
CM_FIQ_STATUS	0x10000060	R	Reset	Core module FIQ status register
CM_FIQ_RSTAT	0x10000064	R	Reset	Core module FIQ raw status register
CM_FIQ_ENSET	0x10000068	R/W	Reset	Core module FIQ enable set register
CM_FIQ_ENCLR	0x1000006C	W	Reset	Core module FIR enable clear register
CM_SPD	0x10000100 to 0x100001FC	R	POR	SDRAM SPD memory

Note

All registers are 32-bits wide and do not support byte writes. Write operations must be word-wide. Bits marked as *reserved* in the following sections must be preserved using read-modify-write operations.

4.3.1 Core module ID register

The core module ID register (CM_ID) is a read-only register that identifies the board manufacturer, board type, and revision.



Copyright © ARM Limited 2001. All rights reserved.

Table 4-3 describes the core module ID register bits.

Table 4-3 CM_ID register bit descriptions

Bits	Name	Access	Function
31:24	MAN	Read	Manufacturer: 0x41 = ARM
23:16	ARCH	Read	Architecture: 0x00 = ASB system bus, ASB processor bus 0x08 = AHB system bus, ASB processor bus
15:12	FPGA	Read	FPGA type: 0x03 = XVC600
11:4	BUILD	Read	Build value (ARM internal use)
3:0	REV	Read	Revision: 0x0 = Rev A 0x1 = Rev B

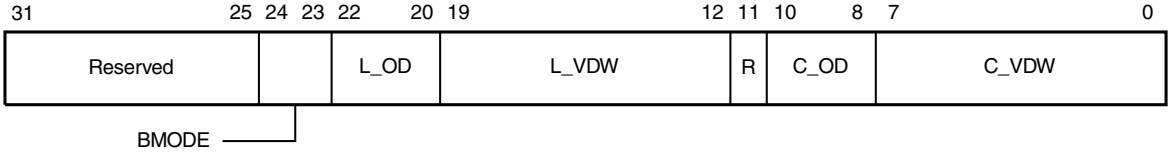
4.3.2 Core module processor identification register

The core module processor register (CM_PROC) is a read-only register that contains the value 0x00000000. This is provided for compatibility with processors that do not have a system control coprocessor (CP15). For the ARM920T, information about the processor can be obtained by reading coprocessor 15 register 0.

Copyright © ARM Limited 2001. All rights reserved.

4.3.3 Core module oscillator register

The core module oscillator register (CM_OSC) is a read/write register that controls the frequency of the clocks generated by the two clock generators (see *Clock generators* on page 3-21). In addition, it provides information about processor bus mode setting.



Note

Before writing to the CM_OSC register, unlock it by writing the value `0x0000A05F` to the CM_LOCK register. After writing the CM_OSC register, relock it by writing any value other than `0x0000A05F` to the CM_LOCK register.

Table 4-4 describes the core module oscillator register bits.

Table 4-4 CM_OSC register

Bits	Name	Access	Function
31:25	Reserved	Use read-modify-write to preserve value.	
24:23	BMODE	Read	This field contains 01 which indicates that the processor bus mode is selected by writing to coprocessor 15 register 1.
22:20	L_OD	Read/write	Memory clock output divider: 000 = divide by 10 001 = divide by 2 (default) 010 = divide by 8 011 = divide by 4 100 = divide by 5 101 = divide by 7 110 = divide by 9 111 = divide by 6.

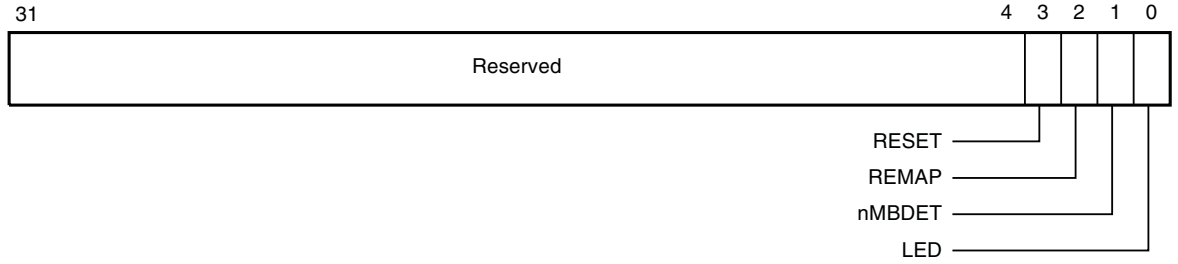
Copyright © ARM Limited 2001. All rights reserved.

Table 4-4 CM_OSC register (continued)

Bits	Name	Access	Function
19:12	L_VDW	Read/write	Processor bus clock VCO divider word. Defines the binary value of the V[7:0] pins of the clock generator (V[8] is tied LOW). `b00100000` = 20MHz (default with OD = 2).
11	Reserved	Use read-modify-write to preserve value.	
10:8	COREOD	Read/write	Core clock output divider: 000 = divide by 10 001 = divide by 2 (default) 010 = divide by 8 011 = divide by 4 100 = divide by 5 101 = divide by 7 110 = divide by 9 111 = divide by 6.
7:0	COREVCO	Read/write	Core clock VCO divider word. Defines the binary value of the V[7:0] pins of the clock generator (V[8] is tied LOW). `b00101010` = 50MHz (default with OD = 2).

4.3.4 Core module control register

The core module control register (CM_CTRL) is a read/write register that provides control of a number of user-configurable features of the core module.



Copyright © ARM Limited 2001. All rights reserved.

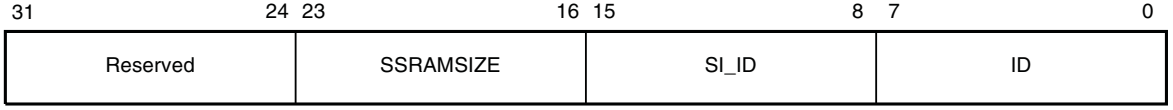
Table 4-5 describes the core module control register bits.

Table 4-5 CM_CTL register

Bits	Name	Access	Function
31:4	Reserved	Use read-modify-write to preserve value.	
3	RESET	Write	This is used to reset the Integrator system. A reset is triggered by writing a 1. Reading this bit always returns a 0 allowing you to use read-modify-write operations without masking the RESET bit.
2	REMAP	Read/write	This only has affect when the core module is mounted onto a motherboard. When this is the case, and this bit is a 0, accesses to the first 256KB (`0x00000000` to `0x0003FFFF`) of memory are redirected into the motherboard (see *Using REMAP* on page 4-3).
1	nMBDET	Read	This bit indicates whether or not the core module is mounted on a motherboard: 0 = mounted on motherboard 1 = standalone.
0	LED	Read/write	This bit controls the green MISC LED on the core module: 0 = LED OFF 1 = LED ON.

4.3.5 Core module status register

The core module status register (CM_STAT) is a read-only register that can be read to determine the SSRAM size, core type, and where in a multi-core module stack this core module is positioned.



Copyright © ARM Limited 2001. All rights reserved.

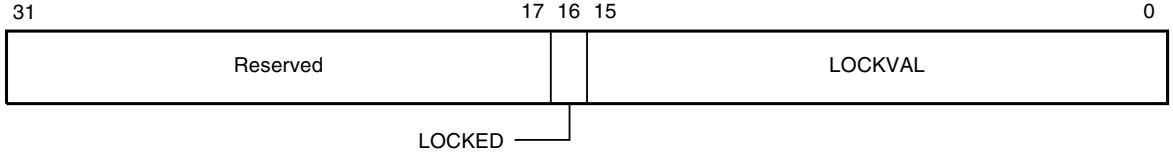
Table 4-6 on page 4-15 describes the core module status register bits.

Table 4-6 CM_STAT register

Bit	Name	Access	Function
31:24	Reserved	Use read-modify-write to preserve value.	
23:16	SSRAMSIZE	Read	SSRAM size. This contains `0x10` to indicate that 1MB is fitted.
15:8	SI_ID	Read	Silicon manufacturer identification. Identifies the manufacturer and type of core fitted to the module: `0x00` = unknown or socket fitted `0x01` = Cirrus (0.25µ, 2V5 core and pads) `0x02` = Panasonic (0.18µ, 1V8 core and 3V3 pads) `0x03` = Samsung (0.25µ, 2V5 core and 3V3 pads) `0x04` - `0xFF` = reserved.
7:0	ID	Read	Card number in stack: `0x00` = core module 0 `0x01` = core module 1 `0x02` = core module 2 `0x03` = core module 3 `0xFF` = invalid or no motherboard attached.

4.3.6 Core module lock register

The core module lock register (CM_LOCK) is a read/write register that is used to control access to the CM_OSC register, allowing it to be locked and unlocked. This mechanism prevents the CM_OSC register from being overwritten accidently.



Copyright © ARM Limited 2001. All rights reserved.

Table 4-7 describes the core module lock register bits.

Table 4-7 CM_LOCK register

Bits	Name	Access	Function
16	LOCKED	Read	This bit indicates if the CM_OSC register is locked or unlocked: 0 = unlocked 1 = locked.
15:0	LOCKVAL	Read/write	Write the value `0x0000A05F` to this register to enable write accesses to the CM_OSC register. Write any other value to this register to lock the CM_OSC register.

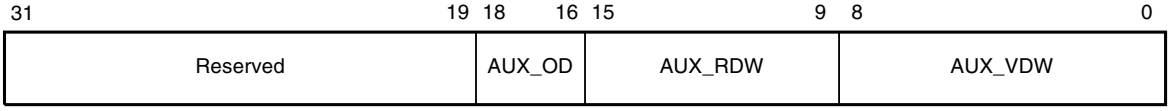
Copyright © ARM Limited 2001. All rights reserved.

4.3.7 Core module local memory bus cycle counter

This register (CM_LMBUSCNT) provides a 16-bit count value. The count increments at the memory bus frequency and can be used as a cycle counter for performance measurement. The register is set to zero by a reset.

4.3.8 Core module auxiliary oscillator register

The core module auxiliary oscillator register (CM_AUXOSC) is a read/write register that controls the frequency of the clock generated by the clock generator for **AUXCLK** (see *Programming the auxiliary clock (AUXCLK)* on page 3-24). This register enables you to set the all three of the control inputs to the clock generator. The default setting of this register gives a 32.369MHz output.



Note

Before writing to the CM_AUXOSC register, unlock it by writing the value `0x0000A05F` to the CM_LOCK register. After writing the CM_AUXOSC register, relock it by writing any value other than `0x0000A05F` to the CM_LOCK register.

Copyright © ARM Limited 2001. All rights reserved.

Table 4-8 describes the core module auxiliary oscillator register bits.

Table 4-8 CM_AUXOSC register

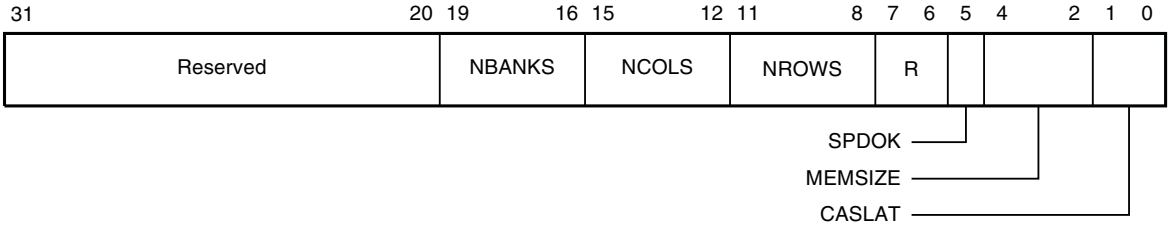
Bits	Name	Access	Function
31:19	Reserved	Use read-modify-write to preserve value.	
18:16	AUX_OD	Read-write	Auxiliary output divider. Sets the binary code on the S[2:0] pins of the clock generator. The divider is encoded as follows: 000 = divide by 0 001 = divide by 2 010 = divide by 8 011 = divide by 4 100 = divide by 5 101 = divide by 7 110 = divide by 9 111 = divide by 6 (default).
15:9	AUX_RDW	Read/write	Auxiliary reference divider word. Defines the binary value on the R[6:0] pins of the clock generator. `b0111111` = 63 (default).
8:0	AUX_VDW	Read/write	Auxiliary clock VCO divider word. Defines the binary value on the V[8:0] pins of the clock generator. `b011111111` = 255 (default).

Copyright © ARM Limited 2001. All rights reserved.

4.3.9 SDRAM status and control register

The SDRAM status and control register (CM_SDRAM) is a read/write register used to set the configuration parameters for the SDRAM DIMM. This control is necessary because of the variety of module sizes and types available.

Writing a value to this register automatically updates the mode register on the SDRAM DIMM.



Note

Before the SDRAM is used it is necessary to read the SPD memory and program the CM_SDRAM register with the parameters indicated in Table 4-9. If these values are not correctly set then SDRAM accesses may be slow or unreliable. See *SDRAM SPD memory* on page 4-27.

Table 4-9 describes the SDRAM status and control register bits.

Table 4-9 CM_SDRAM register

Bits	Name	Access	Function
31:20	Reserved	Use read-modify-write to preserve value.	
19:16	NBANKS	Read/write	Number of SDRAM banks. Set to the same value as byte 5 of SPD EEPROM.
15:12	NCOLS	Read/write	Number of SDRAM columns. Set to the same value as byte 4 of SPD EEPROM.
11:8	NROWS	Read/write	Number of SDRAM rows. Set to the same value as byte 3 of SPD EEPROM.
7:6	Reserved	Use read-modify-write to preserve value.	

Copyright © ARM Limited 2001. All rights reserved.

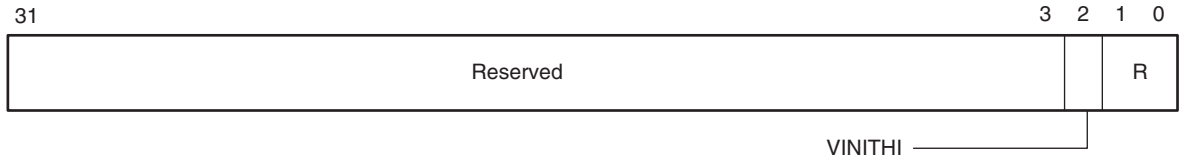
Table 4-9 CM_SDRAM register (continued)

Bits	Name	Access	Function
5	SPDOK	Read	This bit indicates that the automatic copying of the SPD data from the SDRAM module into CM_SPDMEM is complete: 1 = SPD data ready 0 = SPD data not available.
4:2	MEMSIZE	Read/write	These bits specify the size of the SDRAM module fitted to the core module. The bits are encoded as follows: 000 = 16MB 001 = 32MB 010 = 64MB (default) 011 = 128MB 100 = 256MB 101 = Reserved 110 = Reserved 111 = Reserved.
1:0	CASLAT	Read/write	These bits specify the CAS latency set for the core module. The bits are encoded as follows: 00 = Reserved 01 = Reserved 10 = 2 cycles (default) 11 = 3 cycles.

Copyright © ARM Limited 2001. All rights reserved.

4.3.10 Core module initialization register

The core module initialization register (CM_INIT) is used to control the mapping of the vectors test chip.



Note

Before writing to the CM_INIT register, unlock it by writing the value `0x0000A05F` to the CM_LOCK register. After writing the CM_INIT register, relock it by writing any value other than `0x0000A05F` to the CM_LOCK register.

Table 4-10 describes the core module initialization register bits.

Table 4-10 CM_INIT register

Bits	Name	Access	Function
31:3	Reserved	Use read-modify-write to preserve value.	
2	VINITHI	Read/write	This bit controls the state of the VINTHI pin of the ARM920T which sets the initial value of the V bit in coprocessor 15 register 1 and, therefore, the location of the exception vector table. To use high vector mapping, set this bit to 1 and then reset the core module: 0 = vectors at 0 (default after POR) 1 = vectors at `0xFFFF0000`.
1:0	Reserved	Use read-modify-write to preserve value.	

4.3.11 Core module reference clock cycle counter

This register (CM_REFCNT) provides a 32-bit count value. The count increments at the fixed reference clock frequency of 24MHz and can be used as a real-time counter. The register is reset to zero by a reset.

Copyright © ARM Limited 2001. All rights reserved.

4.4 Core module flag registers

The core module flag registers provide you with two 32-bit register locations containing general purpose flags. You can assign any meaning to the flags. The flag registers are listed in Table 4-11 on page 4-22.

Table 4-11 Core module flag registers

Register Name	Address	Access	Reset by	Description
CM_FLAGS	0x10000030	R	Reset	Core module flag register
CM_FLAGSSET	0x10000030	W	Reset	Core module flag set register
CM_FLAGSCLR	0x10000034	W	Reset	Core module flag clear register
CM_NVFLAGS	0x10000038	R	POR	Core module nonvolatile flag register
CM_NVFLAGSSET	0x10000038	W	POR	Core module nonvolatile flag set register
CM_NVFLAGSCLR	0x1000003C	W	POR	Core module nonvolatile flag clear register

The core module provides two distinct types of flag registers:

- the FLAGS registers are cleared by a normal reset, such as a reset caused by pressing the reset button
- the NVFLAGS registers retain their contains after a normal reset and are only cleared by a *Power-On Reset* (POR).

4.4.1 Flag and nonvolatile flag register

The status register contains the current state of the flags.

4.4.2 Flag and nonvolatile flag set register

The flag set locations are used to set bits in the flag registers as follows:

- write 1 to SET the associated flag.
- write 0 to leave the associated flag unchanged.

4.4.3 Flag and nonvolatile flag clear register

The clear locations are used to clear bits in the flag registers as follows:

- write 1 to CLEAR the associated flag
- write 0 to leave the associated flag unchanged

Copyright © ARM Limited 2001. All rights reserved.

4.5 Core module interrupt registers

The core module provides a 3-bit IRQ controller and 3-bit FIQ controller to support the debug communications channel used for passing information between applications software and the debugger. The interrupt control registers are listed in Table 4-12.

Table 4-12 Interrupt controller registers

Register Name	Address	Access	Description
CM_IRQ_STAT	0x10000040	Read	Core module IRQ status register
CM_IRQ_RSTAT	0x10000044	Read	Core module IRQ raw status register
CM_IRQ_ENSET	0x10000048	Read/write	Core module IRQ enable set register
CM_IRQ_ENCLR	0x1000004C	Write	Core module IRQ enable clear register
CM_SOFT_INTSET	0x10000050	Read/write	Core module software interrupt set
CM_SOFT_INTCLR	0x10000054	Write	Core module software interrupt clear
CM_FIQ_STAT	0x10000060	Read	Core module FIQ status register
CM_FIQ_RSTAT	0x10000064	Read	Core module FIQ raw status register
CM_FIQ_ENSET	0x10000068	Read/write	Core module FIQ enable set register
CM_FIQ_ENCLR	0x1000006C	Write	Core module FIR enable clear register

Note

All registers are 32 bits wide and do not support byte writes. Write operations must be word-wide. The values of bits marked as *reserved* in the following sections must be written as 0s.

The IRQ and FIQ controllers each provide three registers for controlling and handling interrupts. These are:

- status register
- raw status register
- enable register, which is accessed using the enable set and enable clear locations.

The way that the interrupt enable, clear, and status bits function for each interrupt is illustrated in Figure 4-4 on page 4-24 and described in the following subsections. The illustration shows the control for one IRQ bit. The remaining IRQ bits and FIQ bits are controlled in a similar way.

Copyright © ARM Limited 2001. All rights reserved.

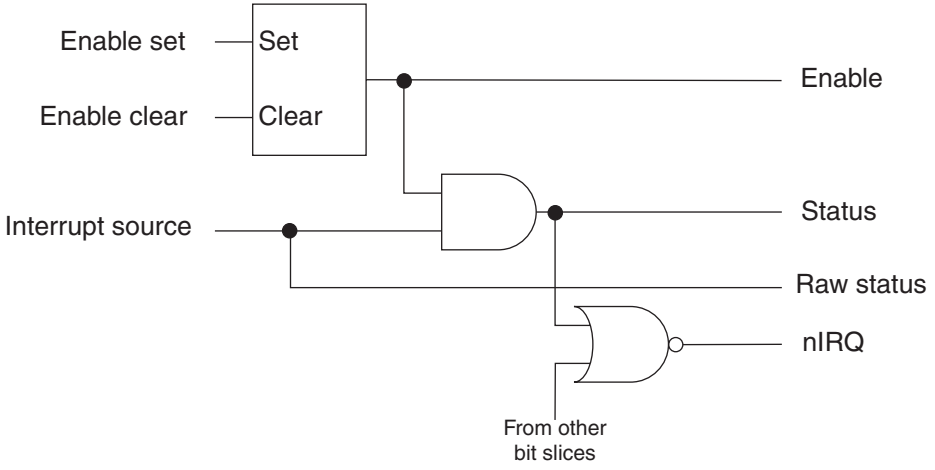


Figure 4-4 Interrupt control

4.5.1 IRQ status and FIQ status registers

The status register contains the logical AND of the bits in the raw status register and the enable register.

4.5.2 IRQ raw status and FIQ raw status registers

The raw status register indicates the signal levels on the interrupt request inputs. A bit set to 1 indicates that the corresponding interrupt request is active.

4.5.3 IRQ and enable set FIQ enable set registers

The enable set locations are used to set bits in the enable registers as follows:

- write 1 to SET the associated bit.
- write 0 to leave the associated bit unchanged.

Read the current state of the enable bits from the ENSET location.

4.5.4 IRQ and enable clear FIQ enable clear registers

The clear set locations are used to set bits in the enable registers as follows:

- write 1 to CLEAR the associated bit
- write 0 to leave the associated bit unchanged

Copyright © ARM Limited 2001. All rights reserved.

4.5.5 Interrupt register bit assignment

The bit assignments for the IRQ and FIQ status, raw status, and enable registers are shown in Table 4-13.

Table 4-13 IRQ and FIQ register bit assignment

Bit	Name	Function
31:3	Reserved	Write as 0. Reads undefined.
2	COMMTx	Debug communications transmit interrupt. This interrupt indicates that the communications channel is available for the processor to pass messages to the debugger.
1	COMMRx	Debug communications receive interrupt. This interrupt indicates to the processor that messages are available for the processor to read.
0	SOFT	Software interrupt

4.5.6 Soft interrupt set and soft interrupt clear registers

The core module interrupt controller provides a register for controlling and clearing software interrupts. This register is accessed using the software interrupt set and software interrupt clear locations. The set and clear locations are used as follows:

- Set the software interrupt by writing to the CM_SOFT_INTSET location:

  write a 1 to SET the software interrupt

  write a 0 to leave the software interrupt unchanged.

- Read the current state of the of the software interrupt register from the CM_SOFT_INTSET location. A bit set to 1 indicates that the corresponding interrupt request is active.

- Clear the software interrupt by writing to the CM_SOFT_INTCLR location:

  write a 1 to CLEAR the software interrupt.

  write a 0 to leave the software interrupt unchanged.

Copyright © ARM Limited 2001. All rights reserved.

The bit assignment for the software interrupt register is shown in Table 4-14.

Table 4-14 IRQ register bit assignment

Bit	Name	Function
31:1	Reserved	Write as 0. Reads undefined.
0	SOFT	Software interrupt.

Note

The *software interrupt* described in this section is used by software to generate IRQs or FIQs. Do not be confuse it with the ARM SWI software interrupt instruction. See the *ARM Architecture Reference Manual*.

Copyright © ARM Limited 2001. All rights reserved.

4.6 SDRAM SPD memory

This area of memory contains a copy of the SPD data from the SPD EEPROM on the DIMM. Because accesses to the EEPROM are very slow, the data is copied to this memory during board initialization to allow faster random access to the SPD data (see *Serial presence detect* on page 3-6). The SPD memory contains 256 bytes of data, the most important of which are as shown in Table 4-15.

Table 4-15 SPD memory contents

Byte	Contents
2	Memory type
3	Number of row addresses
4	Number of column addresses
5	Number of banks
31	Module bank density (MB divided by 4)
18	CAS latencies supported
63	Checksum
64:71	Manufacturer
73:90	Module part number

Check for valid SPD data as follows:

1. Add together all bytes 0 to 62.
2. Logically AND the result with `0xFF`.
3. Compare the result with byte 63.

If the two values match, then the SPD data is valid.

Note

A number of SDRAM DIMMs do not comply with the JEDEC standard and do not implement the checksum byte. The Integrator is not guaranteed to operate with non-compliant DIMMs.

The code segment shown in Example 4-1 on page 4-28 can be used to correctly setup and remap the SDRAM.

Copyright © ARM Limited 2001. All rights reserved.

Example 4-1 SDRAM setup and remap

```
CM_BASE     EQU    0x10000000     ; base address of Core Module registers
SPD_BASE    EQU    0x10000100     ; base address of SPD information

lightled
           ; turn on header LED and remap memory
           LDR    r0, =CM_BASE    ; load register base address
           MOV    r1,#5           ; set remap and led bits
           STR    r1,[r0,#0xc]    ; write the register
           ; setup SDRAM

readspdbit
           ; check SPD bit is set
           LDR    r1,[r0,#0x20]   ; read the status register
           AND    r1,r1,#0x20     ; mask SPD bit (5)
           CMP    r1,#0x20        ; test if set
           BNE    readspdbit      ; branch until the SPD memory has been read

setupsdram
           ; work out the SDRAM size
           LDR    r0, =SPD_BASE  ; point at SPD memory
           LDRB   r1,[r0,#3]     ; number of row address lines
           LDRB   r2,[r0,#4]     ; number of column address lines
           LDRB   r3,[r0,#5]     ; number of banks
           LDRB   r4,[r0,#31]    ; module bank density
           MUL    r5,r4,r3       ; calculate size of SDRAM (MB divided by 4)
           MOV    r5,r5,ASL#2    ; size in MB
           CMP    r5,#0x10       ; is it 16MB?
           BNE    not16          ; if no, move on
           MOV    r6,#0x2        ; store size and CAS latency of 2
           B      writesize

not16
           CMP    r5,#0x20       ; is it 32MB?
           BNE    not32          ; if no, move on
           MOV    r6,#0x6        ; store size and CAS latency of 2
           B      writesize

not32
           CMP    r5,#0x40       ; is it 64MB?
           BNE    not64          ; if no, move on
           MOV    r6,#0xa        ; store size and CAS latency of 2
           B      writesize
```

 Copyright © ARM Limited 2001. All rights reserved.

```
not64
          CMP    r5,#0x80    ; is it 128MB?
          BNE    not128      ; if no, move on
          MOV    r6,#0xe     ; store size and CAS latency of 2
          B      writesize

not128
          ; if it is none of these sizes then it is either 256MB, or
          ; there is no SDRAM fitted so default to 256MB.
          MOV    r6,#0x12    ; store size and CAS latency of 2

writesize
          MOV     r1,r1,ASL#8       ; get row address lines for SDRAM register
          ORR     r2,r1,r2,ASL#12   ; OR in column address lines
          ORR     r3,r2,r3,ASL#16   ; OR in number of banks
          ORR     r6,r6,r3          ; OR in size and CAS latency
          LDR     r0, =CM_BASE      ; point at module registers
          STR     r6,[r0,#0x20]     ; store SDRAM parameters
```


Copyright © ARM Limited 2001. All rights reserved.

 Copyright © ARM Limited 2001. All rights reserved.

Appendix A
Signal Descriptions

This index provides a summary of signals present on the core module main connectors. It contains the following sections:

- *HDRA* on page A-2
- *HDRB* on page A-4
- *Trace connector pinout* on page A-10
- *Logic analyzer connectors* on page A-11.

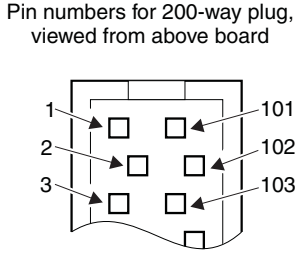
Note

For the Multi-ICE connector pinout and signal descriptions see *JTAG signals* on page 3-30.

Copyright © ARM Limited 2001. All rights reserved.

A.1 HDRA

Figure A-1 shows the pin numbers of the HDRA plug and socket. All pins on the HDRA socket are connected to the corresponding pins on the HDRA plug.



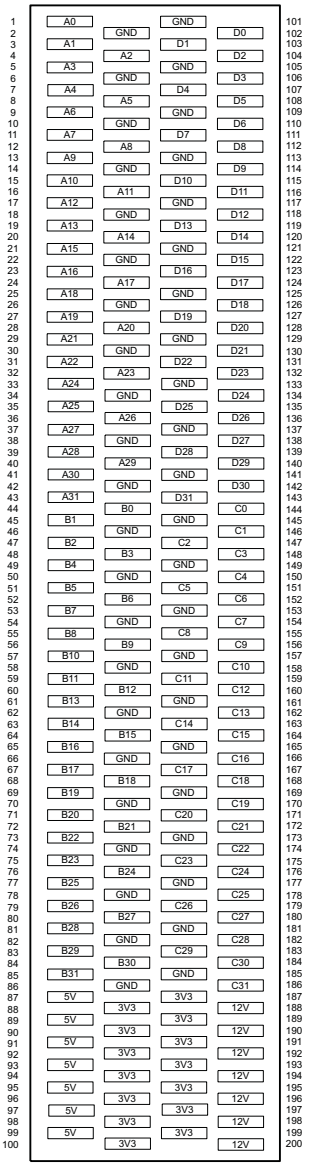


Figure A-1 HDRA plug pin numbering

Copyright © ARM Limited 2001. All rights reserved.

The signals present on the pins labeled A[31:0], B[31:0], and C[31:0] are described in in Table A-1.

Table A-1 Bus bit assignment

Pin label	AHB signal name	ASB signal name	Description
A[31:0]	**HADDR[31:0]**	**BA[31:0]**	System address bus
B[31:0]	Not used	Not used	-
C[31:16]	Not used	Not used	-
C15	**HMASTLOCK**	**BLOK**	Locked transaction
C14	**HRESP1**	**BLAST**	Slave response
C13	**HRESP0**	**BERROR**	Slave response
C12	**HREADY**	**BWAIT**	Slave wait response
C11	**HWRITE**	**BWRITE**	Write transaction
C10	**HPROT2**	Not used	Transaction protection type
C[9:0]	**HPROT[1:0]**	**BPROT[1:0]**	Transaction protection type
C[7:5]	**HBURST[2:0]**	Not used	Transaction burst size
C4	**HPROT[3]**	Not used	Transaction protection type
C[3:2]	**HSIZE[1:0]**	**BSIZE[1:0]**	Transaction width
C[1:0]	**HTRAN[1:0]**	**BTRAN[1:0]**	Transaction type
D[31:0]	**HDATA[31:0]**	Not used	System data bus

Copyright © ARM Limited 2001. All rights reserved.

A.2 HDRB

The HDRB plug and socket have slightly different pinouts, as described below.

A.2.1 HDRB socket pinout

Figure A-2 shows the pin numbers of the socket HDRB on the underside of the core module, viewed from above the core module.

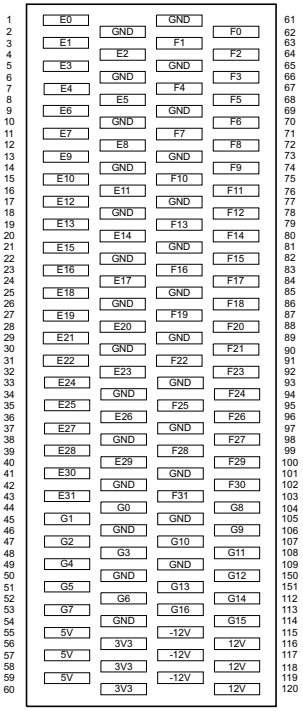


Figure A-2 HDRB socket pin numbering

Copyright © ARM Limited 2001. All rights reserved.

A.2.2 HDRB plug pinout

Figure A-3 shows the pin numbers of the HDRB plug on the top of the core module.

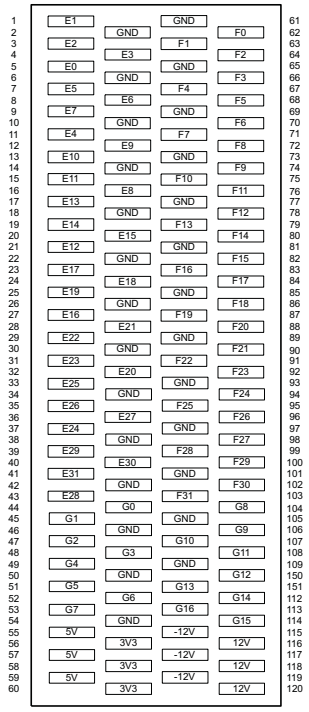


Figure A-3 HDRB plug pin numbering

Copyright © ARM Limited 2001. All rights reserved.

A.2.3 Through-board signal connections

The signals on the pins labeled E[31:0] are cross-connected between the plug and socket so that the signals are rotated through the stack in groups of four. For example, the first block of four are connected as shown in Table A-2.

Table A-2 Signal cross-connections (example)

Plug		Socket
E0	connects to	E1
E1	connects to	E2
E2	connects to	E3
E3	connects to	E0

For details about the signal rotation scheme, see *System bus signal routing* on page 3-16.

The signals on the pins labeled F[31:0] are connected so that pins on the socket are connected to the corresponding pins on the plug.

The signals on G[16:8] and G[5:0] are connected so that pins on the socket are connected to the corresponding pins on the plug.

Pins G[7:6] carry the JTAG **TDI** and **TDO** signals. The signal **TDO** is routed through devices on each board as it passes up through the stack (see *JTAG signals* on page 3-30).

Copyright © ARM Limited 2001. All rights reserved.

A.2.4 HDRB signal descriptions

Table A-3 describes the signals on the pins labeled E[31:0], F[31:0], and G[16:0] for AMBA AHB system bus.

Table A-3 HDRB signal description (AHB)

Pin label	Name	Description
E[31:28]	**SYSCLK[3:0]**	System clock to each core module/expansion card
E[27:24]	**nPPRES[3:0]**	Processor present
E[23:20]	**nIRQ[3:0]**	Interrupt request to processors 3, 2, 1, and 0 respectively
E[19:16]	**nFIQ[3:0]**	Fast interrupt requests to processors 3, 2, 1, and 0 respectively
E[15:12]	**ID[3:0]**	Core module stack position indicator
E[11:8]	**HLOCK[3:0]**	System bus lock from processor 3, 2, 1, and 0 respectively
E[7:4]	**HGRANT[3:0]**	System bus grant to processor 3, 2, 1, and 0 respectively
E[3:0]	**HBUSREQ[3:0]**	System bus request from processors 3, 2, 1, and 0 respectively
F[31:0]	-	Not connected
G16	**nRTCKEN**	RTCK AND gate enable
G[15:14]	**CFGSEL[1:0]**	FPGA configuration select
G13	**nCFGEN**	Sets motherboard into configuration mode
G12	**nSRST**	Multi-ICE reset (open collector)
G11	**FPGADONE**	Indicates when FPGA configuration is complete (open collector)
G10	**RTCK**	Returned JTAG test clock
G9	**nSYSRST**	Buffered system reset
G8	**nTRST**	JTAG reset
G7	**TDO**	JTAG test data out

Copyright © ARM Limited 2001. All rights reserved.

Table A-3 HDRB signal description (AHB) (continued)

Pin label	Name	Description
G6	**TDI**	JTAG test data in
G5	**TMS**	JTAG test mode select
G4	**TCK**	JTAG test clock
G[3:1]	**MASTER[2:0]**	Master ID. Binary encoding of the master currently performing a transfer on the bus. Corresponds to the module ID and to the **HBUSREQ** and **HGRANT** line numbers.
G0	**nMBDET**	Motherboard detect pin

Table A-3 describes the signals on the pins labeled E[31:0], F[31:0], and G[16:0] for AMBA ASB system bus.

Table A-4 HDRB signal description (ASB)

Pin label	Name	Description
E[31:28]	**SYSCLK[3:0]**	System clock to the core module.
E[27:24]	**nPPRES[3:0]**	Processor present.
E[23:20]	**nIRQ[3:0]**	Interrupt request to processor.
E[19:16]	**nFIQ[3:0]**	Fast interrupt requests to processor.
E[15:12]	**ID[3:0]**	Core module stack position indicator.
E[11:8]	Reserved	-
E[7:4]	**SGRNT[3:0]**	System bus grant to processor.
E[3:0]	**SREQ[3:0]**	System bus request from processor.
F[31:0]	-	Not connected.
G16	**nRTCKEN**	**RTCK** AND gate enable.
G[15:14]	**CFGSEL[1:0]**	FPGA configuration select.
G13	**nCFGEN**	Sets motherboard into configuration mode.
G12	**nSRST**	Multi-ICE reset (open collector).
G11	**FPGADONE**	Indicates when FPGA configuration is complete.

Copyright © ARM Limited 2001. All rights reserved.

Table A-4 HDRB signal description (ASB) (continued)

Pin label	Name	Description
G10	**RTCK**	Returned JTAG test clock.
G9	**nSYSRST**	Buffered system reset.
G8	**nTRST**	JTAG reset.
G7	**TDO**	JTAG test data out.
G6	**TDI**	JTAG test data in.
G5	**TMS**	JTAG test mode select.
G4	**TCK**	JTAG test clock.
G[3:1]	**MASTER[2:0]**	Master ID. Binary encoding of the master currently performing a transfer on the bus. Corresponds to the module ID and to the **SREQ** and **SGRANT** line numbers.
G0	**nMBDET**	Motherboard detect pin.

Copyright © ARM Limited 2001. All rights reserved.

A.3 Trace connector pinout

Table A-5 shows the pinout of the Trace connector. The ARM test chip ETMEXTOUT pins are routed to test points.

Table A-5 Trace connector pinout

Channel	Pin	Pin	Channel
no connect	1	2	no connect
no connect	3	4	no connect
GND	5	6	**TRACECLK**
DBGRQ	7	8	**DBGACK**
nSRST	9	10	**EXTTRIG**
TDO	11	12	**VDD** (3.3V)
RTCK	13	14	**VDD** (3.3V)
TCK	15	16	**TRACEPKT7**
TMS	17	18	**TRACEPKT6**
TDI	19	20	**TRACEPKT5**
nTRST	21	22	**TRACEPKT4**
TRACEPKT15	23	24	**TRACEPKT3**
TRACEPKT14	25	26	**TRACEPKT2**
TRACEPKT13	27	28	**TRACEPKT1**
TRACEPKT12	29	30	**TRACEPKT0**
TRACEPKT11	31	32	**TRACESYNC**
TRACEPKT10	33	34	**PIPESTAT2**
TRACEPKT9	35	36	**PIPESTAT1**
TRACEPKT8	37	38	**PIPESTAT0**

The ARM test chip **ETMEXTIN** pins are connected to the FPGA so that they could be driven by a special ETM validation FPGA configuration. In normal configurations the **ETMEXTTRIG** signal from the Trace Port connector is routed through the FPGA to the **ETMEXTIN[1]** pin.

Copyright © ARM Limited 2001. All rights reserved.

A.4 Logic analyzer connectors

A logic analyzer connects to the local memory bus on the core module using high-density AMP Mictor connectors. There are four logic analyzer connectors labelled:

- *Logic analyzer connector BA* on page A-12
- *Logic analyzer connector CONTROL* on page A-13
- *Logic analyzer connector BD* on page A-14
- *Logic analyzer connector SPARE* on page A-15.

The signals names include a prefixed that identifies the source device.

The logic analyzer connection is a high-density AMP Mictor connector. The connectors carries 32 signals and 2 clocks or qualifiers. Figure A-4 shows the connector and identification of pin 1.

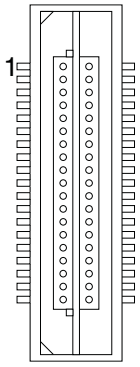


Figure A-4 AMP Mictor connector

Note

Agilent (formerly HP) and Tektronix label these connectors differently, but the assignments of signals to physical pins is appropriate for both systems and pin 1 is always in the same place. The schematic is labelled according to the Agilent pin assignment.

Copyright © ARM Limited 2001. All rights reserved.

A.4.1 Logic analyzer connector BA

Table A-6 shows the pinout of the connector BA.

Table A-6 Connector BA pinout

Channel	Pin	Pin	Channel
no connect	1	2	no connect
GND	3	4	no connect
ARM_BTRAN1	5	6	**LA_BCLK1**
ARM_BA31	7	8	**ARM_BA15**
ARM_BA30	9	10	**ARM_BA14**
ARM_BA29	11	12	**ARM_BA13**
ARM_BA28	13	14	**ARM_BA12**
ARM_BA27	15	16	**ARM_BA11**
ARM_BA26	17	18	**ARM_BA10**
ARM_BA25	19	20	**ARM_BA9**
ARM_BA24	21	22	**ARM_BA8**
ARM_BA23	23	24	**ARM_BA7**
ARM_BA22	25	26	**ARM_BA6**
ARM_BA21	27	28	**ARM_BA5**
ARM_BA20	29	30	**ARM_BA4**
ARM_BA19	31	32	**ARM_BA3**
ARM_BA18	33	34	**ARM_BA2**
ARM_BA17	35	36	**ARM_BA1**
ARM_BA16	37	38	**ARM_BA0**

Copyright © ARM Limited 2001. All rights reserved.

A.4.2 Logic analyzer connector CONTROL

Table A-7 shows the pinout of the CONTROL connector.

Table A-7 Connector CONTROL

Channel	Pin	Pin	Channel
no connect	1	2	no connect
GND	3	4	no connect
nSYSRST	5	6	no connect
SD_nWE	7	8	**SD_nCAS**
SD_CKE1	9	10	**SD_nRAS**
SD_CKE0	11	12	**SRAM_nCKE**
ARM_BnRES	13	14	**SRAM_RnW**
ARM_nIRQ	15	16	**SRAM_nAD**
ARM_nFIQ	17	18	**SRAM_RAMAT0**
SD_nCS0	19	20	**SRAM_nCE**
ARM_BLAST	21	22	**SRAM_nOE**
ARM_BERROR	23	24	**BRIDGE_WRWAIT**
ARM_BWAIT	25	26	**BRIDGE_RDWAIT**
ARM_BLOCK	27	28	**SDRAM_WRWAIT**
ARM_BSIZE1	29	30	**SDRAM_RDWAIT**
ARM_BSIZE0	31	32	**ARM_BURST1**
ARM_BPROT1	33	34	**ARM_BURST0**
ARM_BPROT0	35	36	**ARM_BTRAM1**
ARM_BWRITE	37	38	**ARM_BTRAM0**

Copyright © ARM Limited 2001. All rights reserved.

A.4.3 Logic analyzer connector BD

Table A-8 shows the pinout of the BD connector.

Table A-8 Connector BD pinout

Channel	Pin	Pin	Channel
no connect	1	2	no connect
GND	3	4	no connect
ARM_BWAIT	5	6	**LA_BCLK2**
ARM_BD31	7	8	**ARM_BD15**
ARM_BD30	9	10	**ARM_BD14**
ARM_BD29	11	12	**ARM_BD13**
ARM_BD28	13	14	**ARM_BD12**
ARM_BD27	15	16	**ARM_BD11**
ARM_BD26	17	18	**ARM_BD10**
ARM_BD25	19	20	**ARM_BD9**
ARM_BD24	21	22	**ARM_BD8**
ARM_BD23	23	24	**ARM_BD7**
ARM_BD22	25	26	**ARM_BD6**
ARM_BD21	27	28	**ARM_BD5**
ARM_BD20	29	30	**ARM_BD4**
ARM_BD19	31	32	**ARM_BD3**
ARM_BD18	33	34	**ARM_BD2**
ARM_BD17	35	36	**ARM_BD1**
ARM_BD16	37	38	**ARM_BD0**

Copyright © ARM Limited 2001. All rights reserved.

A.4.4 Logic analyzer connector SPARE

Table A-9 shows the pinout of J11.

Table A-9 HWDATA (J10)

Channel	Pin	Pin	Channel
no connect	1	2	no connect
GND	3	4	no connect
PLD_SPARE22	5	6	no connect
PLD_SPARE21	7	8	PLD_SPARE5
PLD_SPARE20	9	10	PLD_SPARE4
PLD_SPARE19	11	12	**PLD_SPARE3**
PLD_SPARE18	13	14	**PLD_SPARE2**
PLD_SPARE17	15	16	**PLD_SPARE1**
PLD_SPARE16	17	18	**PLD_SPARE0**
PLD_SPARE15	19	20	**ARM_EDBGRQ**
PLD_SPARE14	21	22	**ARM_DBACK**
PLD_SPARE13	23	24	**ARM_DBGEN**
PLD_SPARE12	25	26	**ARM_DEWPT**
PLD_SPARE11	27	28	**ARM_IEBKPT**
PLD_SPARE10	29	30	**ARM_TRACK**
PLD_SPARE9	31	32	**ARM_CHSEX1**
PLD_SPARE8	33	34	**ARM_CHSEX0**
PLD_SPARE7	35	36	**ARM_CHSE1**
PLD_SPARE6	37	38	**ARM_CHSE0**

Copyright © ARM Limited 2001. All rights reserved.

Copyright © ARM Limited 2001. All rights reserved.

Appendix B
Specifications

This appendix contains the specifications for the ARM Integrator/CM920T core module. It contains the following sections:

- *Electrical specification* on page B-2
- *Timing specification* on page B-3
- *Mechanical details* on page B-11.

Copyright © ARM Limited 2001. All rights reserved.

B.1 Electrical specification

This section provides details of the voltage and current characteristics for the core module.

B.1.1 Bus interface characteristics

Table B-1 shows the core module electrical characteristics for the system bus interface. The core module uses 3.3V and 5V sources. The 12V inputs are supplied by the motherboard but not used by the core module.

Table B-1 Core module electrical characteristics

Symbol	Description	Min	Max	Unit
3V3	Supply voltage (interface signals)	3.1	3.5	V
5V	Supply voltage	4.75	5.25	V
V_{IH}	High-level input voltage	2.0	3.6	V
V_{IL}	Low-level input voltage	0	0.8	V
V_{OH}	High-level output voltage	2.4	-	V
V_{OL}	Low-level output voltage	-	0.4	V
C_{IN}	Input capacitance	-	20	pF

B.1.2 Current requirements

Table B-2 shows the current requirements at room temperature and nominal voltage. These measurements include the current drawn by Multi-ICE, which is approximately 160mA at 3.3V.

Table B-2 Current requirements

System	At 3.3V	At 5V
Standalone core module	1A	100mA
Motherboard and one core module	1.5A	500mA

An Integrator/AP with additional core or logic modules draws more current, and future core modules may require more current. For these reasons, provision is made to power the system with an ATX-type power supply.

Copyright © ARM Limited 2001. All rights reserved.

B.2 Timing specification

This section is a reference for designers adding modules on to an Integrator system. The timing information presented here is representative only. Specific modules and FPGA revisions will deviate from these numbers, but they provide some guidance when constraining FPGA designs.

The following sections detail the timing parameters for a typical ASB and AHB modules and motherboards.

B.2.1 Integrator timing parameters and the AMBA Specification

The parameters listed are those specified in the AMBA Specification with the following important differences:

- only output valid and input setup times are quoted
- the required input hold time (Tih) is always less than or equal to 0ns
- the output hold time (Toh) is always greater than 2ns.

Each version and revision of the FPGA has subtly different timing. The figures are those you can expect under nominal conditions and should be used as a guideline when designing you own motherboards and modules. The figures have been rounded to simplify timing analysis and constraints.

B.2.2 AHB system bus timing parameters

Table B-3 shows the clock and reset timing parameters.

Table B-3 Clock and reset parameters

Parameter	Description	Time (ns)	Notes
Tclk	**HCLK** minimum clock period	30	Representative of worst case maximum frequency
Tisrst	**HRESETn** deasserted setup time before HCLK	15	Applies to modules only
Tovrst	**HRESETn** deasserted valid time before **HCLK**	15	Applies only to the module or motherboard implementing the reset source

Copyright © ARM Limited 2001. All rights reserved.

Table B-4 shows the AHB slave input parameters.

Table B-4 AHB slave input parameters

Parameter	Description	Time (ns)	Notes
Tissel	**HSELx** setup time before **HCLK**	n/a	**HSELx** are internally generated, not visible at the pins
Tistr	Transfer type setup time before **HCLK**	5	
Tisa	**HADDR[31:0]** setup time before **HCLK**	10	
Tisctl	**HWRITE**, **HSIZE[2:0]** and **HBURST[2:0]** control signal setup time before **HCLK**	5	
Tiswd	Write data setup time before **HCLK**	5	
Tisrdy	Ready setup time before **HCLK**	5	
Tismst	Master number setup time before **HCLK** (SPLIT-capable only)	n/a	Applies to modules with split capable slaves only
Tismlck	Master locked setup time before **HCLK** (SPLIT-capable only)	n/a	Applies to modules with split capable slaves only

Table B-5 shows the AHB slave output parameters.

Table B-5 AHB slave output parameters

Parameter	Description	Time (ns)	Notes
Tovrsp	Response valid time after **HCLK**	15	
Tovrd	Data valid time after **HCLK**	15	
Tovrdy	Ready valid time after **HCLK**	15	
Tovsplt	Split valid time after **HCLK** (SPLIT-capable only)	n/a	Applies to modules with split capable slaves only

Copyright © ARM Limited 2001. All rights reserved.

Table B-6 shows the bus master input timing parameters.

Table B-6 Bus master input timing parameters

Parameter	Description	Time (ns)	Notes
Tisgnt	**HGRANTx** setup time before **HCLK**	5	Modules implementing masters only
Tisrdy	Ready setup time before **HCLK**	5	
Tisrsp	Response setup time before **HCLK**	5	
Tisrd	Read data setup time before **HCLK**	5	

Table B-7 shows bus master output timing parameters.

Table B-7 Bus master ouput timing paramaters

Parameter	Description	Time (ns)	Notes
Tovtr	Transfer type valid time after **HCLK**	15	
Tova	Address valid time after **HCLK**	15	
Tovctl	Control signal valid time after **HCLK**	15	
Tovwd	Write data valid time after **HCLK**	15	
Tovreq	Request valid time after **HCLK**	15	Modules implementing masters only
Tovlck	Lock valid time after **HCLK**	15	Modules implementing masters only

Table B-8 shows the AHB arbiter input parameters. Applies only to the module or motherboard implementing the arbiter

Table B-8 AHB arbiter input parameters

Parameter	Description	Time (ns)
Tisrdy	Ready setup time before **HCLK**	5
Tisrsp	Response setup time before **HCLK**	5
Tisreq	Request setup time before **HCLK**	10
Tislck	Lock setup time before **HCLK**	10

Copyright © ARM Limited 2001. All rights reserved.

Table B-8 AHB arbiter input parameters

Parameter	Description	Time (ns)
Tissplt	Split setup time before **HCLK**	10
Tistr	Transfer type setup time before **HCLK**	5
Tisctl	Control signal setup time before **HCLK**	5

Table B-9 shows the AHB arbiter output parameters. Applies only to the module or motherboard implementing the arbiter

Table B-9 AHB arbiter output parameters

Parameter	Description	Time (ns)
Tovgnt	Grant valid time after **HCLK**	15
Tovmst	Master number valid time after **HCLK**	15
Tovmlck	Master locked valid time after **HCLK**	15

B.2.3 ASB system bus timing parameters

Table B-10 shows the clock and reset parameters.

Table B-10 Clock and reset parameters

Parameter	Description	Time (ns)	Notes
Tclk	**BCLK** minimum clock period	40	Representative of worst case maximum frequency
Tclkl	**BCLK** LOW time	20	
Tclkh	**BCLK** HIGH time	20	
Tisnres	**BnRES** deasserted setup to rising **BCLK**	15	Applies to modules only
Tovnres	**BnRES** deasserted valid after rising **BCLK**	15	Applies only to the module or motherboard implementing the reset source

Copyright © ARM Limited 2001. All rights reserved.

Table B-11 shows the ASB slave input parameters. Applies only to the module or motherboard implementing the arbiter.

Table B-11 ASB slave input paramaters

Parameter	Description	Time (ns)	Notes
Tisdsel	**DSEL** setup to falling **BCLK**	n/a	**DSEL** is internally generated, not visible at the pins
Tisa	**BA[31:0]** setup to falling **BCLK**	10	Path through decoder is up to 30ns
Tisctl	**BWRITE** and **BSIZE[1:0]** setup to falling **BCLK**	10	
Tisdw	For write transfers, **BD[31:0]** setup to falling **BCLK**	10	

Table B-12 shows the ASB slave output parameters.

Table B-12 ASB slave output parameters

Parameter	Description	Time (ns)
Tovresp	**BWAIT**, **BERROR** and **BLAST** valid after falling **BCLK**	10
Tovdr	For read transfers, **BD[31:0]** valid after rising **BCLK**	30

Table B-13 shows the bus master input timing parameters

Table B-13 Bus master input parameters

Parameter	Description	Time (ns)	
Tisresp	**BWAIT**, **BERROR** and **BLAST** setup to rising **BCLK**	15	
Tisdr	For read transfers, **BD[31:0]** setup to falling **BCLK**	10	
Tisagnt	**AGNT** setup to rising **BCLK**	10	Modules implementing masters only

Copyright © ARM Limited 2001. All rights reserved.

Table B-14 shows the bus master output timing parameters.

Table B-14 Bus master output parameters

Parameter	Description	Time (ns)
Tovtr	BTRAN valid after rising BCLK	10
Tova	BA[31:0] valid after rising BCLK, all transfer types	10
Tovctl	BWRITE, BSIZE[1:0] and BPROT[1:0] valid after rising BCLK, all transfer types	10
Tovdw	BD[31:0] valid after rising BCLK, all transfer types	30
Tovlok	BLOK valid after rising BCLK	10
Tovareq	AREQ valid after rising BCLK	10

Table B-15 shows the ASB decoder input parameters.

Table B-15 ASB decoder input parameters

Parameter	Description	Time (ns)
Tistr	**BTRAN** setup to falling **BCLK**	10
Tisresp	**BWAIT**, **BERROR** and **BLAST** setup to rising **BCLK**	15

Table B-16 ASB decoder output parameters.

Table B-16 ASB decoder output parameters

Parameter	Description	Time (ns)	Notes
Tovresp	**BWAIT**, **BERROR** and **BLAST** valid after falling **BCLK**	10	
Tovdsel	**DSEL** valid after rising **BCLK**	n/a	**DSEL** is internally generated, not visible at the pins

Copyright © ARM Limited 2001. All rights reserved.

Table B-17 shows the ASB arbiter input parameters.Applies only to the module or motherboard implementing the arbiter..

Table B-17 ASB arbiter input parameters

Parameter	Description	Time (ns)
Tisareq	**AREQ** setup to falling **BCLK**	10
Tisresp	**BWAIT** setup to rising **BCLK**	10

Table B-18 ASB arbiter output parameters. applies only to the module or motherboard implementing the arbiter.

Table B-18 ASB arbiter output parameters

Parameter	Description	Time (ns)
Tovagnt	**AGNT** valid after falling **BCLK**	10

Table B-19 shows the ASB arbiter combinatorial parameters.

Table B-19 ASB arbiter combinatorial parameters

Parameter	Description	Time (ns)	Notes
Tlokagnt	Delay from valid **BLOK** to valid **AGNT**	n/a	Not applicable, arbiter samples all inputs

B.2.4 Notes on FPGA timing analysis

The system bus on all Integrator boards is routed between FPGAs. These FPGAs are routed with timing constraints like those shown in the table in *AHB system bus timing parameters* on page B-3 and *ASB system bus timing parameters* on page B-6. The exact performance of a system depends on the timing parameters of the motherboard and all modules in the system. Some allowance also needs to be made for clock skew, routing delays and number of modules (that is, loading).

Not all FPGAs will meet the ideal timing parameters, due to the complexity of the design or routing congestion within the device. For this reason the PLL clock generators on Integrator default to a safe low value that all modules can achieve.

A detailed timing analysis involves calculating the input/output delays between modules for all parameters. In general, a simpler approach is to increase the operating frequency until the system becomes unstable. The maximum stable operating frequency for your board combination is likely to be a few MHz lower.

Copyright © ARM Limited 2001. All rights reserved.

ARM processors and core module FPGAs do not dissipate large amounts of heat. However, to be sure of stable operation, run the test program for a few minutes. Experiments show that the FPGAs, when operating at maximum system bus frequency, slowly increase in temperature, but that the maximum is typically less than 35°C.

Copyright © ARM Limited 2001. All rights reserved.

B.3 Mechanical details

The core module is designed to be stackable on a number of different motherboards. Its size allows it to be mounted onto a CompactPCI motherboard while allowing the motherboard to be installed in a card cage.

Figure B-1 shows the mechanical outline of the core module.

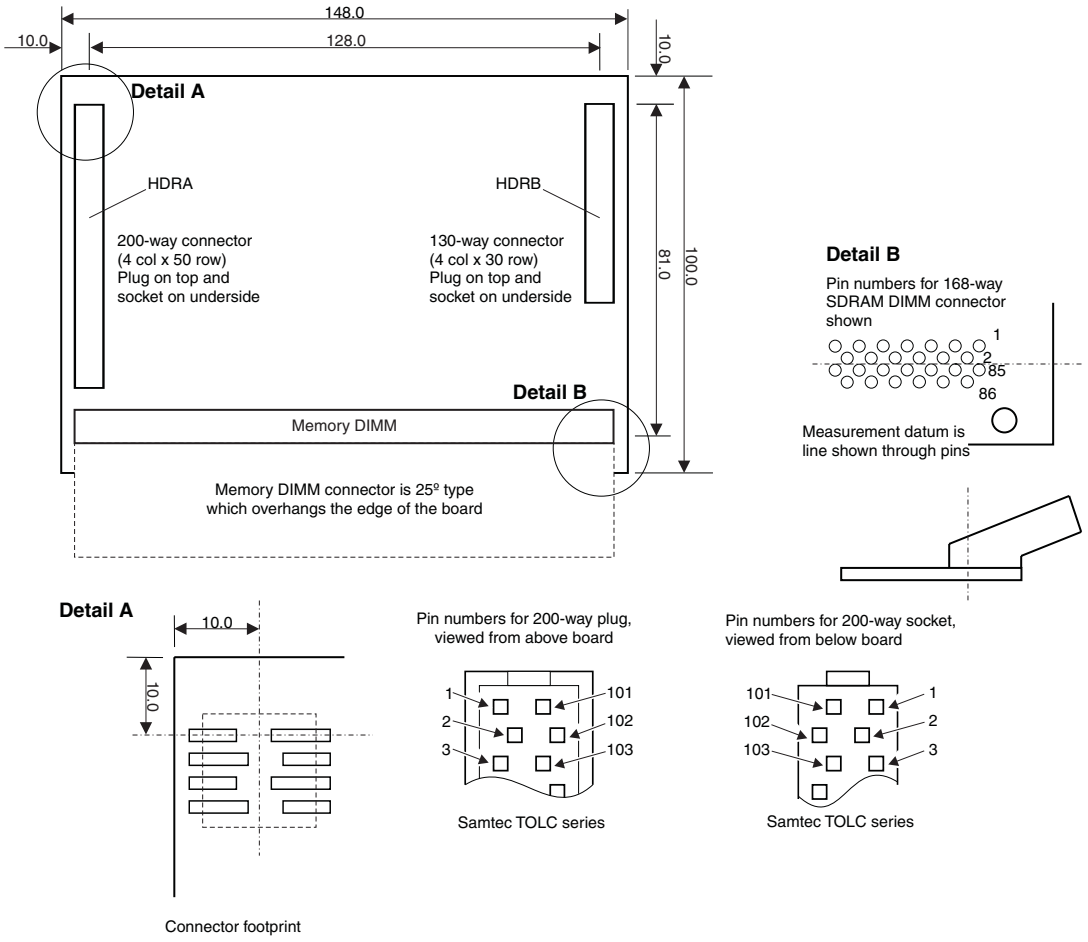


Figure B-1 Board outline

Copyright © ARM Limited 2001. All rights reserved.

Copyright © ARM Limited 2001. All rights reserved.

Index

The items in this index are listed in alphabetical order, with symbols and numerics appearing at the end. The references given are to page numbers.

A

About this book
  feedback xi
Access arbitration, SDRAM 3-6
Accesses
  boot ROM 4-3
  SSRAM 4-3
Accesses SDRAM 4-5
Address decoding, module 3-19
Alias SDRAM addresses 4-6
ARM processor, overview 1-5
Assmbled Integrator system 2-5
AUX oscillator register 4-17
AUX_OD bits 4-18
AUX_RDW bits 4-18
AUX_VCW bits 4-18

B

Block diagram 1-4
Boot ROM, accesses 4-3
Bus bridge, system 3-11
Bus operating modes 3-18

C

Calculating the CORECLK speed 3-22
CAS latency, setting 4-20
CE Declaration of Conformity ii
Checking for valid SPD data 4-27
Chip selects, EBI 3-5
Clock generator 1-6, 3-21
Clock, reference 3-25
CM_AUXOSC register 4-17
CM_CTL register 4-3
CM_CTRL register 4-13
CM_FIQ_ENCLR register 4-23
CM_FIQ_ENSET register 4-23
CM_FIQ_RSTAT register 4-23
CM_FIQ_STAT register 4-23
CM_ID Register 4-10
CM_ID register 4-7
CM_INIT register 4-21
CM_IRQ_ENCLR register 4-23
CM_IRQ_ENSET register 4-23
CM_IRQ_RSTAT register 4-23
CM_IRQ_STAT register 4-23
CM_LMBUSCNT register 4-17
CM_LOCK register 4-15, 4-21
CM_OSC register 3-22, 3-23, 4-12
CM_PROC register 4-11
CM_REFCNT register 4-21
CM_SDRAM 3-7
CM_SDRAM register 4-19
CM_SOFT_INTCLR register 4-23
CM_SOFT_INTSET register 4-23
CM_SPD 3-7
CM_STAT register 4-14
CONFIG LED 3-26
CONFIG link 3-26
Configuration mode 3-29
Connecting Multi-ICE 2-4
Connecting power 2-3
Connector
  Logic analyzer A-11
Connectors

Copyright © ARM Limited 2001. All rights reserved.

HDRA and HDRB 1-3
Multi-ICE 2-4
power 2-3
Controller
clock 3-21
reset 1-5, 3-8
SDRAM 1-5, 3-6
SSRAM 3-3
Controllers
clock 1-6
FIQ 4-23
IRQ 4-23
Core module control register 4-13
Core module FPGA 1-5
Core module ID 2-6
ID selection 3-19
ID signals 3-19
Core module local memory bus cycle counter 4-17
Core module registers 4-9
Core module, stack position 4-14
CORECLK 3-22

D

Debug comms channel 4-23
Debugging modes 3-29
DIMM socket 1-3

E

EBI chip selects 3-5
Electrical characteristics B-2
Enable register, flag 4-22
Enable register, interrupt 4-24
Ensuring safety 1-12
Exception vector mapping 4-8

F

FIFOs 3-11
FIQ controller 4-23
Fitting SDRAM 2-2
Flag registers 4-22
FPGA 1-5

H

HDRA 3-16
HDRA and HDRB connectors 1-3
HDRA pinout A-2
HDRB plug pinout A-5
HDRB signals A-7, A-8
HDRB socket pinout A-4

I

ID bits, Core module ID, reading 4-15
ID, core module 2-6
Interrupt control 4-24
Interrupt register bit assignment 4-25
Interrupt registers 4-23
Interrupt, signal routing 3-20
Interupt status 4-24
IRQ and FIQ register bit assignment 4-25
IRQ controller 4-23

J

JTAG 3-26
JTAG debug 1-7
JTAG scan path 3-27
JTAG signals 3-30
JTAG, connecting 2-4

L

Local SDRAM 4-5
Location of connectors 1-3
Lock register 4-15
Logic analyzer connector A-11

M

MBDET bit 4-14
Memory
volatile 1-6
Memory map 4-2
MEMSIZE 4-20
Microprocessor core 3-2
MISC LED control 4-14
Module ID selection 3-19
Motherboard, attaching the core module 2-5
Multi-ICE 1-7, 3-26
connecting 2-4
Multi-ICE connector 1-3

N

nEPRES signals 3-19
nMBDET signal 3-28
Normal debug mode 3-29
nPRES signals 3-19

O

Operating mode, SDRAM 3-6
Oscillator register 4-12
Output divider 3-22, 3-24

P

Pinout
HDRA A-2
HDRB plug A-5
HDRB socket A-4
Trace A-10
Power connector 1-3, 2-3
Powering an attached core module 2-6
Precautions 1-12
Preventing damage 1-12
Processor core clock 3-22
Processor register 4-11
Product feedback xi

R

Raw status register, interrupt 4-24
REFCLK 3-21, 3-25
Reference clock 3-25
Register addresses 4-9
Registers 1-5
CM_AUXOSC 4-17
CM_CTL 4-3

Copyright © ARM Limited 2001. All rights reserved.

CM_CTRL 4-13
CM_FIQ_ENCLR 4-23
CM_FIQ_ENSET 4-23
CM_FIQ_RSTAT 4-23
CM_FIQ_STAT 4-23
CM_ID 4-7, 4-10
CM_INIT 4-21
CM_IRQ_ENCLR 4-23
CM_IRQ_ENSET 4-23
CM_IRQ_RSTAT 4-23
CM_IRQ_STAT 4-23
CM_LMBUSCNT 4-17
CM_LOCK 4-15
CM_OSC 3-22, 3-23, 4-12
CM_PROC 4-11
CM_REFCNT 4-21
CM_SDRAM 4-19
CM_SOFT_INTCLR 4-23
CM_SOFT_INTSET 4-23
CM_STAT 4-14
Related publications x
REMAP bit 4-14
Remap, effect of 4-3
Reset control bit 4-14
Reset controller 1-5, 3-8

S

SDRAM
fitting 2-2
operating without 2-2
SDRAM access arbitration 3-6
SDRAM accesses 4-5
SDRAM controller 1-5, 3-6
SDRAM operating mode 3-6
SDRAM repeat mapping 4-5
SDRAM status and control register 4-19
SDRAM, SPD memory 4-27
Serial presence detect 3-6
Setting CAS latency 4-20
Setting SDRAM size 4-20
Setup
power connections 2-3
standalone 2-2
Signal routing, interrupts 3-20
Signals nEPRES 3-19
signals nPRES 3-19
SI_ID bits 4-15
Software interrupt registers 4-25
Software reset 4-14
SPD memory 4-27
SPDOK bit 4-20
SSRAM accesses 4-3
SSRAM controller 3-3
SSRAMSIZE bits 4-15
Standalone core module 2-2
Status and configuration registers 1-5
Status register 4-14
Status register, flag 4-22
Status register, interrupt 4-24
Supplying power 2-3
System architecture 1-4
System bus
operating modes 3-18
System bus bridge 1-5, 3-11
System bus signal routing 3-16

T

TDI signal 3-28
Test chip, overview 1-5
Through-board signals A-6
Trace connector pinout A-10

U

Using the core module with a motherboard 2-5

V

VCO divider 3-22, 3-24
Vector mapping 4-8
VINITHI bit 4-21
Volatile memory 1-6

Numerics

Signals
ID 3-19

Copyright © ARM Limited 2001. All rights reserved.

Copyright © ARM Limited 2001. All rights reserved.